Universität des Saarlandes
Fakultät für Mathematik und Informatik
Fachbereich Informatik

Cryptographic Techniques for Privacy and Access Control in Cloud-Based Applications

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

von
Manuel Reinert

Saarbrücken, Januar 2018

Tag des Kolloquiums:	29. Juni 2018
Dekan:	Prof. Dr. Sebastian Hack

Prüfungsausschuss:

Vorsitzender:	Prof. Dr. Christoph Sorge
Berichterstattende:	Prof. Dr. Matteo Maffei
	Prof. Dr. Aniket Kate
Akademische Mitarbeiterin:	Dr. Fabienne Eigner

Zusammenfassung

Die Digitalisierung ist eine der größten Herausforderungen für Industrie und Gesellschaft. Neben vielen Geschäftsbereichen betrifft diese auch, insbesondere sensible, Nutzerdaten. Daher sollten persönliche Informationen so gut wie möglich gesichert werden. Zugleich brauchen Cloud-basierte Software-Anwendungen, die der Nutzer verwenden möchte, Zugang zu diesen Daten.

Diese Dissertation fokussiert sich auf das sichere Auslagern und Teilen von Daten unter Wahrung der Privatsphäre, auf das Abfragen von geschützten, ausgelagerten Daten und auf die Nutzung persönlicher Informationen als Zugangsberechtigung für erpressungsresistente Bewertungsplattformen. Zu diesen drei Themen präsentieren wir kryptographische Techniken und Protokolle, die den Stand der Technik voran treiben. Der erste Teil stellt Multi-Client Oblivious RAM (ORAM) vor, das ORAM durch die Möglichkeit, Daten unter Wahrung von Vertraulichkeit und Integrität mit anderen Nutzern zu teilen, erweitert. Der zweite Teil befasst sich mit Freuquency-hiding Order-preserving Encryption. Wir zeigen, dass dem Stand der Technik eine formale Betrachtung fehlt, was zu Angriffen führt. Um Abhilfe zu schaffen, verbessern wir die Sicherheitsdefinition und beweisen, dass das existierende Verschlüsselungsschema diese durch minimale Änderung erfüllt. Abschließend entwickeln wir ein erpressungsresistentes Bewertungsportal. Erpressungsresistenz wurde bisher hauptsächlich im Kontext von elektronischen Wahlen betrachtet.

Abstract

Digitization is one of the key challenges for today's industries and society. It affects more and more business areas and also user data and, in particular, sensitive information. Due to its sensitivity, it is important to treat personal information as secure and private as possible yet enabling cloud-based software to use that information when requested by the user.

In this thesis, we focus on the privacy-preserving outsourcing and sharing of data, the querying of outsourced protected data, and the usage of personal information as an access control mechanism for rating platforms, which should be protected from coercion attacks. In those three categories, we present cryptographic techniques and protocols that push the state of the art. In particular, we first present multi-client oblivious RAM (ORAM), which augments standard ORAM with selective data sharing through access control, confidentiality, and integrity. Second, we investigate on recent work in frequency-hiding order-preserving encryption and show that the state of the art misses rigorous treatment, allowing for simple attacks against the security of the existing scheme. As a remedy, we show how to fix the security definition and that the existing scheme, slightly adapted, fulfills it. Finally, we design and develop a coercion-resistant rating platform. Coercion-resistance has been dealt with mainly in the context of electronic voting yet also affects other areas of digital life such as rating platforms.

Preface

This dissertation builds on the following research results. The author contributed to all of them as main author as well as to their elaboration.

Chapter 2 builds on the following works:

- Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Brief Announcement: Towards Security and Privacy for Outsourced Data in the Multi-Party Setting. In *Proceedings of the Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'14)*. 2014, ACM. [139]
- Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Privacy and Access Control for Outsourced Personal Records. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'15)*. 2015, IEEE Press. [141]
- Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Maliciously Secure Multi-Client ORAM. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'17)*. 2017, Springer. [142]

Chapter 3 builds on the following work:

- Matteo Maffei, Manuel Reinert, and Dominique Schröder. On the Security of Frequency-Hiding Order-Preserving Encryption. In *Proceedings of the International Conference on Cryptology and Network Security (CANS'17)*. 2017, Springer. [145]

Chapter 4 builds on the following work:

- Matteo Maffei, Kim Pecina, Manuel Reinert, and Ahmed Salem. CR^{2}P: Coercion-Resistant Rating Platforms. 2017 [144]

Acknowledgments

I would like to thank a number of people without the help and support of whom the completion of this thesis would not have been possible.

To begin with, I thank my supervisor Matteo Maffei for persuading me into becoming a PhD student in his group and pursuing the research path rather than going to industry right after finishing my Master's thesis with him. Once started, his work just began in teaching me all the skills necessary to write this thesis, let alone his ability to push my strengths while encouraging me to, at the same time, work on my weaknesses. Not only sometimes did he have to endure long discussions. Matteo, I am deeply grateful for the past six years, in which you taught me a lot, not only as a researcher, but also as a friend.

I wish to thank Aniket Kate, who agreed to take the time to write a second review for this thesis.

I am grateful to Matteo Maffei, Giulio Malavolta, Kim Pecina, Dominique Schröder and Ahmed Salem for fruitful collaborations on interesting topics, great discussions, and a nice working atmosphere. Special thanks go to Dominique Schröder for always believing in me, reminding me from time to time that situations are not always as bad as they seem, and being a mentor and friend.

Thanks go to my colleagues Fabienne Eigner, Sebastian Gerling, Niklas Grimm, Ilya Grishchenko, Giulio Malavolta, Pedro Moreno-Sanchez, Kim Pecina, Tim Ruffing, and Clara Schneidewind, who made the time of being a PhD student an enjoyable and often funny one.

Special thanks go to my friends Lara Schneider and Hazem Torfah, who both went the entire path from Bachelor to Master to PhD studies together with me.

Last but not least, I am deeply grateful to my big family, who offered the non-scientific support necessary to complete this thesis. Be it in times of joy or in those of frustration, they supported me and always found a way to build me up again when I was down.

To *Winfried*, RIP

Contents

1. Introduction

1.1. Digitization of User Records

Digitization keeps finding its way into almost every area of life. Not only is digitization tremendously changing the way companies and businesses manufacture products and offer services, it also affects end users and customers alike. One prominent example for this change is the medical sector. While healthcare personnel previously kept paper-based records for every patient, for a couple of years now, sometimes even decades, they have been moving to electronic records. In hospitals, we often find hybrid approaches [130] consisting of a general electronic health record, which is maintained centrally by the hospital's administration, and paper-based notes and lab results, which are used in the daily treatment of a patient.

The goal of a digitization of patient records is to facilitate the sharing of health information among doctors, hospitals, insurances, and other healthcare personnel. This goal is clearly motivated by an improved and holistic treatment of the patient that takes all available information into account. Different countries try to reach the goal using different approaches, ranging from centralized systems that store medication data in a nationwide database (e.g., in Austria[1]) to hybrid or distributed systems (e.g., the electronic health card in Germany). It is easy to see that especially central databases ease the treatment of patients. Additionally, it makes the study of medical diseases more efficient since more information is available for querying.

Regardless of the way in which a record is stored, the patient is equipped with several rights concerning the record. For instance, in the US, the Health Insurance Portability and Accountability Act (HIPAA) [101] allows patients to access their record, get a copy of it, get information about how and with whom information is shared, and to perform further actions. Likewise, the Social Act (Sozialgesetzbuch) [180] in Germany regulates the usage of the electronic health card (elektronische Gesundheitskarte, eGK [75]): the patient first has to opt in and can later on decide who may access which information on the card. She can also access certain information herself at home. Similar regulations that tend into the same direction also exist for other countries, for instance, Austria [86] and Canada [151].

Besides the question of how and where medical data is stored and processed, the e-health domain is target of other services that aim at improving the quality of service. For instance, rating platforms such as RateMDs[2] or Jameda[3] allow patients to rate their

[1]https://elga.gv.at
[2]https://www.ratemds.com
[3]https://www.jameda.de

doctors and read the reviews of other patients so as to decide to which doctor or specialist to go for a certain medical problem.

1.2. Security and Privacy Problems of Digitization

Digitizing all records and making them available to qualified personnel for processing and retrieval as well as letting their owners use them raises several security and privacy problems. We discuss those problems in the following section using the e-health example.

Outsourced storage of personal records. Giving a patient the control over her health record eases the enforcement of existing law, which says that the patient should be able to select with whom to share information and that she should have access to her medical data. In order to make the data accessible to healthcare personnel, however, there are two options: the first is to hand out cards to users, as done in Germany, and store the record on that card. Besides having several advantages, there are two central disadvantages: first, if the card is lost or the patient does not carry it along with her, then the necessary information is not available in case of an emergency. Second, most of those cards can only store a very limited amount of data, say 64 kilobytes, which is far from fitting an X-ray image, for example. Hence, storing information *only* on the card does not seem to be a good idea, which is why it must be combined with a storage system for redundancy and availability of the necessary information.

The second option is thus to only use the redundant storage or a personal health record (health record controlled by the patient). We focus on an architecture that places the health information in the cloud. Clearly, letting the record reside in the cloud and giving different individuals access to certain information implies several problems. The first problem is related to *selective access control*, for instance, only the doctor should be able to write a medical bill for a health insurance while only a pharmacy should be able to read a prescription so as to serve the patient with the respective medicine. We refer to those two concepts in the rest of the thesis often with the terms *integrity* and *secrecy*. The second problem targets the storage provider. One option is to trust the cloud/storage provider; but this trust is hard to motivate given the sensitivity of the stored data and the value of such data for businesses. Consequently, the data must be concealed from the cloud provider. Moreover, it has been shown that, knowing the structure of data, encrypting the data does not suffice to keep it secret [107]. Using the access pattern of users, one can sometimes learn the exact content of the data, even if it is encrypted. This holds true in particular for medical data such as DNA-sequences. The structure of such a sequence is known and an attacker can infer a symptom or disease given the areas of the sequence that are accessed by an analyst. A property to protect against such abuse is called *obliviousness*, which we consider a central and desirable feature of the outsourced storage of high-sensitivity data such as medical data.

Secure querying of outsourced records. In the realm of a central database that contains medical data of all patients and might be accessed by researchers or medical analysts, we are confronted with similar problems. The company that hosts the data should not learn the exact data but at the same time it should be able to answer search queries efficiently. Efficiency and obliviousness are often in conflict with each other. Since medical data resides in the realm of what we nowadays call “big data”, the central administration is more likely focusing on efficiency rather than perfect obliviousness. Consequently, the goal here is to protect the data as much as possible yet allowing for search queries at a speed that is not debilitatingly slow.

Using personal records for authentication in rating platforms. Rating platforms, independent of the domain, are very often consulted by users to improve their confidence in upcoming decisions [150]. In order to be useful, rating platforms must be trustworthy. Only if a user can trust the correctness and faithfulness of a rating, she will be able to base decisions on that. Conversely, the question arises whether the rating platform host should be fully trusted. The answer to that question is no since platform managers might have economic interests to show specific ratings that highlight the good aspects of a doctor rather than showing the bad reviews on the top of a page. Today, due to the importance of ratings for decision making, entire businesses exists solely for the purpose of rating management and getting the best out of user generated content (e.g., reevoo[4] and bazaarvoice[5]). On the other side of the market there exist single individuals and organizations that sell fake reviews, advertised on websites such as fiverr.[6]

So in order to keep the rating process as independent as possible from the interests of the person or object being rated, we cannot put full trust in the platform host. The minimum amount of security that we want to achieve for rating platforms is *access control*, which allows only qualified users to post a rating, and *verifiability*, which allows a reader of a rating to verify whether it originates from a qualified user or not. Qualified here means that only a patient who visited a doctor is eligible to write a review for that doctor. This minimalistic requirement is, however, not sufficient to express the desired property: in order to allow raters to be entirely honest, they must be *anonymous* such that disadvantageous consequences in case of a negative review are ruled out. Moreover, being able to authenticate somewhere in an anonymous fashion paves the way for *coercion attacks*, which have been common, especially for doctor ratings in the past [164] in form of fake reviews and review suppression. Our security and privacy goals for rating platforms include thus authorization, anonymity, coercion-resistance, and verifiability.

1.3. Our Contribution

This thesis contributes to each of the three areas and problems described above, the secure and privacy-preserving access to outsourced storage, the secure querying of outsourced encrypted databases, as well as the rating of objects, persons, or institutions in a secure and privacy-preserving fashion. In order to target the problems described above, we present cryptographic protocols along with formal models and efficient realizations.

[4]`https://www.reevoo.com`
[5]`http://www.bazaarvoice.com`
[6]`https://www.fiverr.com`

1.3.1. Privacy and Access Control for Outsourced Storage

Outsourcing storage securely is a well-studied problem that has attracted lots of research in the past. Typically, security in that scenario is falsely connoted with encryption of the data. Unfortunately, as described above, depending on the kind of data that is encrypted, it is sometimes not enough to just encrypt since the access patterns to the encrypted data reveal their underlying plaintexts.

The state-of-the-art technology to counter this attack is called oblivious RAM (ORAM) [88]. It achieves access pattern hiding on top of encrypting the data and is supported by an active line of research [3,8,29,41,65,66,91,106,136,138,159,170,178,183–185]. ORAM has originally been proposed to protect the cells in a random-access-memory, which are accessed by the CPU during computation, from a curious observer. Consequently, ORAM considers only a single client (originally the CPU) and a single server (originally the RAM). This restricted setting, however, diminishes the usability of ORAM in a multi-client setting, which is required in order to allow a data owner to selectively share data with other clients. Clearly, ORAM only supports a trivial form of sharing: all-or-nothing, i.e., either one gives a client access to the entire database, by handing out the secret ORAM key, or one does not give it access to any data.

Since single-client ORAM is not satisfactory for sharing outsourced information, in the first part of this thesis, we present a multi-client ORAM framework that overcomes the data-sharing and access control problems existing with single-client ORAM solutions. The framework is called Group ORAM. We formally define all relevant security and privacy properties in a unified attacker model. The most important notions are *secrecy*, meaning that only authorized clients may read certain data, *integrity*, meaning that only authorized clients may write certain data, *tamper-resistance*, meaning that in case the server is malicious, an honest client cannot be convinced of the correctness of a maliciously modified data entry, and *obliviousness*, meaning that different, but equally long access patterns to the database are indistinguishable from each other.

Obliviousness comes in two flavors, which require fundamentally different technical approaches: if the server is not allowed to collude with malicious clients, then we can use the classical techniques used to achieve obliviousness in the single-client setting. If, however, the server may collude with malicious clients, then single-client ORAM solutions are no longer sufficient. In that case, stronger primitives are necessary to achieve the required security level. Based on the framework, we present five different multi-client ORAM constructions, which vary in the properties they achieve and in how efficient they are. In order to demonstrate those differences with respect to efficiency, we present a performance evaluation that demonstrates the feasibility of the strongest constructions as well as the practicality of the most efficient construction.

1.3.2. Secure Querying of Outsourced Databases

Querying encrypted data can be accomplished in two ways. The first one is to use techniques such as private information retrieval [53], which are expensive in terms of computational effort both on client and server side and in terms of communication. The second approach is to use specific encryption schemes which allow server-aided querying such as order-preserving encryption [179]. Unfortunately, numerous attacks exist that show that the used encryption schemes do not provide security in practice when confronted with auxiliary information that is easily available, such as census data or information contained in telephone books.

One of the central problems in order-preserving encryption is the frequency leakage of underlying plaintexts: if the same plaintext is encrypted multiple times, this is reflected in the ciphertexts. A possible solution to this problem is frequency-hiding order-preserving encryption, which attempts to prevent exactly that kind of leakage. However, the only existing scheme to date [115] has shown to be vulnerable against attacks based on auxiliary information [96] and the theoretical treatment of the new notion is still in the fledgling stage.

In the second part of this thesis, we further investigate the theoretical aspects of frequency-hiding order-preserving encryption and achieve interesting but somewhat surprising results: we first show that the definition presented by Kerschbaum [115] is imprecise and its natural interpretation leads to severe attacks that break the security of the scheme. Going one step further, we can even show that the definition itself is not achievable at all by any order-preserving encryption scheme. Second, we identify the imprecision in the definition and pinpoint to the problem in the security proof. In fact, the security proof contains the answer to the problem of devising an achievable definition: it makes additional assumptions that are not reflected in the definition. Hence, we devise a more rigorous definition that is achievable. Finally, we show that a slight adaption of Kerschbaum's scheme provably achieves the more precise definition.

1.3.3. Coercion-Resistant Rating Platforms

Product ratings are a key asset for marketing strategies and opinion-forming, since a majority of online users trusts them as far as decision making is concerned. Ratings help users to decide which product to buy next or which doctor to visit for a special kind of symptom [150]. The popularity and power of ratings open, at the same time, the door for fake or bought reviews or ratings, which are hard to detect and even harder to protect against. Furthermore, rating mechanisms often fall prey to forced abstention, for instance, a doctor might let her patient sign a form in which she relinquishes to ever review that doctor on any platform [164].

Existing and deployed rating platforms do not protect against coercion attacks due to a very simple reason: knowing the username and password suffices to gain access to the user's reviews and provide the attacker with the possibility to post reviews himself. Hence, in order to protect users against coercion attacks, we have to think fundamentally different with respect to authorization, which must be both anonymous and serve as eligibility proof for the platform provider, telling that the user is authorized to rate a certain product. Additional to that, the platform and users together need a way to signalize coercion so that the platform can successfully drop coerced ratings in a way that does not tell the coercer that it has happened. Finally, the result of the rating process must be verifiable so as to enhance the trustworthiness of the entire system.

In the third part of this thesis, we present a coercion-resistant rating platform. For authorization policies that target the eligibility of the rating and the anonymity of the rater, the platform leverages attribute-based credentials. For instance, if a user is only allowed to rate a product if she has bought it (cf. Amazon verified purchase [5]), then the vendor might hand out a credential to the user indicating that she has bought that product on a certain date at a certain store. The user can then use that credential, hide all unnecessary information (e.g., the store where the product has been purchased), and authenticate at the rating platform provider so as to provide evidence for the eligibility of her review. In order to prevent coercion attacks, the platform establishes a novel primitive called coercion-resistant identification token between itself and the user. The token authenticates the user to the platform in an anonymous fashion and can be faked in case of coercion so that the coercer is equipped with an invalid token, which he cannot detect. The platform, receiving a rating accompanied by an invalid token then discards that rating. Finally, a tallying protocol inspired by the electronic voting literature serves the purpose of making the process publicly verifiable. The key challenge lies in designing such a tallying protocol without violating any of the aforementioned properties and still guaranteeing the correctness of the rating process.

1.4. Outline

The main part of this thesis consists of three parts. In the first part we present our results on privacy and access control for outsourced data in a data sharing scenario (Chapter 2). The second part contains our investigations on the security of outsourced data encrypted with frequency-hiding order-preserving encryption (Chapter 3). In the third part, we present our coercion-resistant rating platform (Chapter 4). Finally, we conclude this thesis (Chapter 5).

In both the first and the third part we formally define the considered model or scenario. We then move to our cryptographic constructions and prove them secure according to the defined model. We finish with performance evaluations and concluding remarks. The second part also defines the considered model and presents attacks in that model on the state of the art. We then present an impossibility result for the given definitions and fix them, so that they can be achieved.

Part I
Privacy and Access Control for Outsourced Storage

We are very good at hiding from ourselves what we do not want to know.

Terry Pratchett, Unseen Academicals

2. Group ORAM for Privacy and Access Control in Outsourced Personal Records

Cloud storage has rapidly become a cornerstone of many IT infrastructures, constituting a seamless solution for the backup, synchronization, and sharing of large amounts of data. Putting user data in the direct control of cloud service providers, however, raises security and privacy concerns related to the integrity of outsourced data, the accidental or intentional leakage of sensitive information, the profiling of user activities and so on. Furthermore, even if the cloud provider is trusted, users having access to outsourced files might be malicious and misbehave. These concerns are particularly serious in sensitive applications like personal health records and credit score systems.

We tackle this problem in this chapter. We present Π_{GORAM}, a definitional framework for Group Oblivious RAM, in which we formalize several security and privacy properties such as secrecy, integrity, anonymity, and obliviousness. Π_{GORAM} allows per entry access control, as selected by the data owner. Π_{GORAM} is the first framework to define such a wide range of security and privacy properties for outsourced storage. Regarding obliviousness, we tackle two different attacker models: our first definition protects against an honest-but-curious server while our second definition protects against such a server colluding with malicious clients.

In the latter model, we prove a server-side computational lower bound of $\Omega(n)$, i.e., every operation requires to process a constant fraction of the database. Furthermore, we present two constructions: a pure cryptographic instantiation, which achieves an $O(\sqrt{N})$ amortized communication and computation complexity and a construction based on a trusted proxy with logarithmic communication and server-side computational complexity. The second construction bypasses the previously established lower bound. Both schemes achieve secrecy, integrity, and obliviousness with respect to a server colluding with malicious clients, but not anonymity due to the deployed access control mechanism.

In the former model, we present a cryptographic system that achieves secrecy, integrity, obliviousness, and anonymity. In the process of designing an efficient construction, we developed three new, generally applicable cryptographic schemes, namely, batched zero-knowledge proof of shuffle correctness, the hash-and-proof paradigm, which even improves upon the former, and an accountability technique based on chameleon signatures, which we consider of independent interest.

We implemented our constructions in Amazon Elastic Compute Cloud (EC2) and ran a performance evaluation demonstrating the scalability and efficiency of our construction.

2.1. Introduction

Cloud storage has rapidly gained a central role in the digital society, serving as a building block of consumer-oriented applications (e.g, Dropbox, Microsoft SkyDrive, and Google Drive) as well as particularly sensitive IT infrastructures, such as personal record management systems. For instance, credit score systems rely on credit bureaus (e.g., Experian, Equifax, and TransUnion in US) collecting and storing information about the financial status of users, which is then made available upon request. As a further example, personal health records (PHRs) are more and more managed and accessed through web services (e.g., private products like Microsoft HealthVault and PatientsLikeMe in US and national services like ELGA in Austria), since this makes PHRs readily accessible in case of emergency even without the physical presence of the e-health card and eases their synchronization across different hospitals.

Despite its convenience and popularity, cloud storage poses a number of security and privacy issues. The first problem is related to the *secrecy* of user data, which are often sensitive (e.g., PHRs give a complete picture of the health status of citizens) and, thus, should be concealed from the server. A crucial point to stress is that preventing the server from reading user data (e.g., through encryption) is necessary but not sufficient to protect the privacy of user data. Indeed, as shown in the literature [107, 159], the capability to link consecutive accesses to the same file can be exploited by the server to learn sensitive information: for instance, it has been shown that the access patterns to a DNA sequence allow for determining the patient's disease. Hence the *obliviousness* of data accesses is another fundamental property for sensitive IT infrastructures: the server should not be able to tell whether two consecutive accesses concern the same data or not, nor to determine the nature of such accesses (read or write). Furthermore, the server has in principle the possibility to modify client's data, which can be harmful for several reasons: for instance, it could drop data to save storage space or modify data to influence the statistics about the dataset (e.g., in order to justify higher insurance fees or taxes). Therefore another property that should be guaranteed is the *integrity* of user data.

Finally, it is often necessary to share outsourced documents with other clients, yet in a controlled manner, i.e., selectively granting them read and write permissions: for instance, PHRs are selectively shared with the doctor before a medical treatment and a prescription is shared with the pharmacy in order to buy a medicine. *Data sharing* complicates the enforcement of secrecy and integrity properties, which have to be guaranteed not only against a malicious server but also against malicious clients. Notice that the simultaneous enforcement of these properties is particularly challenging, since some of them are in seeming contradiction. For instance, *access control* seems to be incompatible with the obliviousness property: if the server is not supposed to learn which file the client is accessing, how can he check that the client has the rights to do so?

2.1.1. Our Contributions

In this chapter, we present Π_{GORAM}, a novel framework for privacy-preserving cloud-storage. Users can share outsourced data with other clients, selectively granting them read and write permissions, and verify the integrity of such data. These are hidden from the server and access patterns are oblivious. Π_{GORAM} is the first system to achieve such a wide range of security and privacy properties for storage outsourcing. More specifically, the contributions of this chapter are the following:

- We formalize the problem statement by introducing the notion of Group Oblivious RAM (Group ORAM) in Section 2.2. Group ORAM extends the concept of Oblivious RAM [88] (ORAM) [1] by considering multiple, possibly malicious clients, with read and/or write access to outsourced data, as opposed to a single client. We propose a formal security model that covers a variety of security and privacy properties, such as data integrity, data secrecy, obliviousness of access patterns, and anonymity. For obliviousness, we consider two variants: first, we define obliviousness against malicious clients. Intuitively, none should be able to determine which entry is read by which client. However, write operations are oblivious only with respect to the server and to those clients who cannot read the modified entry, since clients with read access can obviously notice that the entry has changed. Second, we define obliviousness only against a malicious server that does not collude with clients. We find this slightly weaker obliviousness definition interesting, in particular, because it allows for significantly more efficient cryptographic constructions.

- In the obliviousness against malicious clients setting, we establish an insightful *computational lower bound* (Section 2.3): if clients have *direct access* to the database, the number of operations *on the server side* has to be linear in the database size. Intuitively, the reason is that if a client does not want to access all entries in a read operation, then it must know where the required entry is located in the database. Since malicious clients can share this information with the server, the server can determine for each read operation performed by an honest client, which among the entries the adversary has access to might be the subject of the read, and which certainly not.

- We present PIR-GORAM, the first *cryptographic construction* that ensures the obliviousness of data accesses against malicious clients as well as access control (Section 2.4). Our construction relies on Private Information Retrieval (PIR) [53] to achieve obliviousness, uses a new accumulation technique based on an oblivious gossiping protocol to reduce the communication in an amortized fashion, and combines public-key cryptography and zero-knowledge proofs for access control.

- To bypass the aforementioned lower bound, we consider the recently proposed *proxy-based setting* [29, 136, 170, 184, 193], which assumes the presence of a trusted party

[1]ORAM is a technique originally devised to protect the access pattern of software on the local memory and then used to hide the data and the user's access pattern in storage outsourcing services.

mediating the accesses between clients and server. We show, in particular, that TAO-GORAM, a simple variant of TaoStore [170], guarantees obliviousness in the malicious setting as well as access control (Section 2.5).

- We then move to the setting in which obliviousness holds only against a malicious server who does not collude with clients. We first introduce GORAM, a cryptographic instantiation based on a novel combination of ORAM [187], predicate encryption [113], and zero-knowledge (ZK) proofs (of shuffle correctness) [22, 94] (Section 2.6). This construction is secure, but building on off-the-shelf cryptographic primitives is not practical. In particular, clients prove to the server that the operations performed on the database are correct through ZK proofs of shuffle correctness, which are expensive when the entries to be shuffled are tuples of data, as opposed to single entries.

- As a first step towards a practical instantiation, we maintain the general design of GORAM, but we replace the expensive ZK proofs of shuffle correctness with a new proof technique called *batched* ZK proofs of shuffle correctness (Section 2.6.5.1). A batched ZK proof of shuffle correctness significantly reduces the number of ZK proofs by "batching" several instances and verifying them together. Since this technique is generically applicable in any setting where one is interested to perform a zero-knowledge proof of shuffle correctness over a list of entries, each of them consisting of a tuple of encrypted blocks, we believe that it is of independent interest. This second realization greatly outperforms the first solution and is suitable for databases with relatively small entries, accessed by a few users, but it does not scale to large entries and many users.

- In a second step to improve upon the previous variant of GORAM, we present a novel technique based on universal pair-wise hash functions [44] that speeds up batched shuffle proofs even further (Section 2.6.5.2). As opposed to batched shuffle proofs, which are secure when repeated λ many times where λ is the security parameter, this construction computes a constant number of proofs. It is generally applicable and we show that it outperforms batched shuffle proofs by one order of magnitude.

- To obtain a scalable solution, we explore some trade-offs between security and efficiency. First, we present A-GORAM, which relies on a new accountability technique based on chameleon signatures (Section 2.7). The idea is to let clients perform arbitrary operations on the database, letting them verify each other's operation a-posteriori and giving them the possibility to blame misbehaving parties. Secondly, in S-GORAM, we replace the relatively expensive predicate encryption, which enables sophisticated role-based and attribute-based access control policies, with the more efficient broadcast encryption, which suffices to enforce per-user read/write permissions, as required in the personal record management systems we consider (Section 2.8). This approach leads to a very efficient solution that scales to large files and thousands of users, with a combined communication-computation overhead of only 7% (resp. 8%) with respect to state-of-the-art, single-client ORAM constructions for reading (resp. writing) on a 1GB storage with 1MB block size (for larger datasets or block sizes, the overhead is even lower).

We have implemented our constructions and conducted a performance evaluation demonstrating the scalability and efficiency of our constructions (Section 2.10). Although Group ORAM is generically applicable, the large spectrum of security and privacy properties, as well as the efficiency and scalability of the system, make it particularly suitable for the management of large amounts of sensitive data, such as personal records. We exemplify that in a case study presented in Section 2.11.

2.2. Definitional Framework

We detail the problem statement by formalizing the concept of Group ORAM (Section 2.2.1), introducing the attacker model (Section 2.2.2), and presenting the security and privacy properties (Section 2.2.3).

2.2.1. Group ORAM

We consider a data owner $\mathcal{O}$ outsourcing her database $\mathcal{DB} = d_1, \ldots, d_m$ to the server $\mathcal{S}$. A set of clients $\mathcal{C}_1, \ldots, \mathcal{C}_n$ can access parts of the database, as specified by the access control policy set by $\mathcal{O}$. This is formalized as an n-by-m matrix $\mathbf{AC}$ (where $|\mathbf{AC}|_\mathsf{r} = n$ and $|\mathbf{AC}|_\mathsf{c} = m$ denote the number of rows and columns, respectively), defining the permissions of the clients on the files in the database: $\mathbf{AC}(i, j)$ (i.e., the j-th entry of the i-th row) denotes the access mode for client i on data d_j. An entry in $\mathbf{AC}$ can take one of the values $\perp$ (no access), R (read access), or RW (read-write access).

At registration time, each client $\mathcal{C}_i$ receives a capability cap_i, which gives $\mathcal{C}_i$ access to $\mathcal{DB}$ as specified in the corresponding row of $\mathbf{AC}$. Furthermore, we assume the existence of a capability $cap_\mathcal{O}$, which grants permissions for all of the operations that can be executed by the data owner only.

In the following we formally characterize the notion of Group ORAM. Intuitively, a Group ORAM is a collection of two algorithms and four interactive protocols, used to setup the database, add clients, add an entry to the database, change the access permissions to an entry, read an entry, and overwrite an entry. In the sequel, we let $\langle A, B \rangle$ denote a protocol between the PPT machines A and B, $|\mathbf{a}|$ the length of the vector $\mathbf{a}$ of access modes, and $\mathbf{a}(i)$ the element at position i in $\mathbf{a}$. In all our protocols $|\mathcal{DB}| = |\mathbf{AC}|_\mathsf{c}$ and we write $\mathbf{AC} \,||_\mathsf{r}\, \mathbf{a}$ and $\mathbf{AC} \,||_\mathsf{c}\, \mathbf{a}$ to denote row and column concatenation, respectively.

Definition 2.1 (Group ORAM). *A* Group ORAM *scheme is a tuple of (interactive)* PPT *algorithms* $\Pi_{\mathsf{GORAM}} = (\mathsf{gen}, \mathsf{addCl}, \mathsf{addE}, \mathsf{chMode}, \mathsf{read}, \mathsf{write})$*, such that:*

$(cap_\mathcal{O}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda, n)$: *on input a security parameter λ and an integer n indicating the maximum number of clients, this algorithm outputs $(cap_\mathcal{O}, \mathcal{DB})$ where $\mathcal{DB} := [\,]$ is the database and $cap_\mathcal{O}$ is the data owner's capability. Additionally, the algorithm initializes the access control matrix $\mathbf{AC} := [\,]$, which is treated as a global variable hereafter.*

$\{cap_i, \mathsf{deny}\} \leftarrow \mathsf{addCl}(cap_{\mathcal{O}}, \mathbf{a})$: *on input the data owner's capability* $cap_{\mathcal{O}}$ *and an access permission vector* $\mathbf{a}$*, this algorithm checks whether* $|\mathbf{a}| = |\mathbf{AC}|_{\mathsf{c}}$*. In that case,* $\mathbf{AC} = \mathbf{AC}\,||_{\mathsf{r}}\,\mathbf{a}$ *and the algorithm outputs a client capability* cap_i *that grants access permissions to* $\mathcal{C}_i$ *according to* $\mathbf{a}$*. Otherwise, it outputs* deny.

$\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$: *on input the data owner's capability* $cap_{\mathcal{O}}$*, an access permission vector* $\mathbf{a}$*, and a data* d*, this protocol checks whether* $|\mathbf{a}| = |\mathbf{AC}|_{\mathsf{r}}$*. In that case,* $\mathbf{AC} = \mathbf{AC}\,||_{\mathsf{c}}\,\mathbf{a}$ *and the algorithm outputs a database* $\mathcal{DB}'$*, which is equal to* $\mathcal{DB}$ *augmented by* d*, granting clients access permissions according to* $\mathbf{a}$*. Otherwise, it outputs* deny.

$\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$: *on input the data owner's capability* $cap_{\mathcal{O}}$*, an access permission vector* $\mathbf{a}$*, and an index* j*, this protocol changes the access permissions for the* j*-th entry as specified by* $\mathbf{a}$*. If* $j \leq |\mathcal{DB}|$ *and* $|\mathbf{a}| = |\mathbf{AC}|_{\mathsf{r}}$*, then the* j*-th column of* $\mathbf{AC}$ *is replaced by* $\mathbf{a}$*.*

$\{d, \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$: *on input a capability* cap_i *and an index* j*, this protocol either outputs* $d := \mathcal{DB}(j)$ *or* deny *if* $|\mathcal{DB}| < j$ *or* $\mathbf{AC}(i, j) = \bot$*.*

$\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle$: *on input a capability* cap_i*, an index* j*, and a data* d*, this protocol overwrites the* j*-th entry of* $\mathcal{DB}$ *with* d*. It succeeds and outputs* $\mathcal{DB}'$ *if and only if* $\mathbf{AC}(i, j) = \mathsf{RW}$*, otherwise it outputs* deny.

2.2.2. The Attacker Model

We consider an adversarial model in which the data owner $\mathcal{O}$ is honest, the clients $\mathcal{C}_1, \ldots, \mathcal{C}_n$ may be malicious, and the server $\mathcal{S}$ is assumed to be honest-but-curious (HbC)[2] (and not to collude with clients). These assumptions are common in the literature (see, e.g., [4, 51]) and are well justified in a cloud setting, since it is of paramount importance for service providers to keep a good reputation, which discourages them from visibly misbehaving, while they may have an incentive in passively gathering sensitive information given the commercial interest of personal data.

Although we could limit ourselves to reason about all security and privacy properties in this attacker model, we find it interesting to state and prove some of them even in a stronger attacker model, where the server can arbitrarily misbehave and collude with malicious clients. This allows us to characterize which properties unconditionally hold true in our different systems, i.e., even if the server is compromised (cf. the discussion in the end of this section).

2.2.3. Security and Privacy Properties

2.2.3.1. Adversary interfaces

We define all security and privacy properties as game-based definitions and, thus, need to provide an interface to the adversary that allows him to talk to the game challenger.

[2] I.e., the server is regarded as a passive adversary, following the protocol but seeking to gather additional information

```
$\mathsf{Setup}(1^\lambda, x)$
$(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$
$\mathbf{AC} = [\,]$
$\mathsf{CAP} = \emptyset$
$Cor = \emptyset$
if $\neg x$ then
  return $(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor)$
$\mathcal{DB}' = [\,]$
return $(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor, \mathcal{DB}')$
```

```
$\mathcal{O}_{\mathsf{corCl}}(i)$
if $\exists y \in \mathsf{CAP}, z.\ y = (i, z)$ then
  $Cor = Cor \cup i$
  return $z$
```

```
$\mathcal{O}^x_{\mathsf{chMode}}(\mathbf{a}, j)$
if $x$ then
  $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$
  $d \leftarrow \mathcal{DB}(j)$
  $\mathcal{DB}' = \mathcal{DB}'[j \mapsto d]$
else
  $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{A}(\mathcal{DB})\rangle$
```

```
$\mathcal{O}_{\mathsf{query}}(b, i_0, j_0, d_0, i_1, j_1, d_1)$
if $j_0 \leq |\mathcal{DB}| \wedge j_1 \leq |\mathcal{DB}| \wedge$
  $\exists y_0 \in \mathsf{CAP}, y_1 \in \mathsf{CAP}, z_0, z_1.$
    $y_0 = (i_0, z_0) \wedge y_1 = (i_1, z_1)$ then
  if $(d_0 = \bot \implies \mathbf{AC}(i_0, j_0) \neq \bot)$
    $\wedge\ (d_0 \neq \bot \implies \mathbf{AC}(i_0, j_0) = \mathsf{RW}$
      $\wedge\ \forall i \in Cor.\ \mathbf{AC}(i, j_0) = \bot)$
    $\wedge\ (d_1 = \bot \implies \mathbf{AC}(i_1, j_1) \neq \bot)$
    $\wedge\ (d_1 \neq \bot \implies \mathbf{AC}(i_1, j_1) = \mathsf{RW}$
      $\wedge\ \forall i \in Cor.\ \mathbf{AC}(i, j_1) = \bot)$ then
    if $d_b = \bot$ then
      $d \leftarrow \langle \mathcal{C}_{\mathsf{read}}(z_b, j_b), \mathcal{A}(\mathcal{DB})\rangle$
    else
      $\langle \mathcal{C}_{\mathsf{write}}(z_b, j_b, d_b), \mathcal{A}(\mathcal{DB})\rangle$
```

```
$\mathcal{O}^x_{\mathsf{addE}}(\mathbf{a}, d)$
if $x$ then
  $y \leftarrow \langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$
  if $y \neq \mathsf{deny}$ then
    $\mathcal{DB} = y$
    $\mathcal{DB}' = \mathcal{DB}' \| d$
else
  $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{A}(\mathcal{DB})\rangle$
  if $\exists \mathcal{DB}'$ then
    $\mathcal{DB}' = \mathcal{DB}' \| d$
```

```
$\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$
$cap_i \leftarrow \mathsf{addCl}(cap_{\mathcal{O}}, \mathbf{a})$
$\mathsf{CAP} \leftarrow \mathsf{CAP} \cup \{(i, cap_i)\}$
```

```
$\mathcal{O}^x_{\mathsf{read}}(i, j)$
if $\exists y, \in \mathsf{CAP}, z.\ y = (i, z)$ then
  if $x$ then
    if $i \in Cor$ then
      $\langle \mathcal{A}(z, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$
    else
      $d \leftarrow \langle \mathcal{C}_{\mathsf{read}}(z, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$
  else
    $d \leftarrow \langle \mathcal{C}_{\mathsf{read}}(z, j), \mathcal{A}(\mathcal{DB})\rangle$
```

```
$\mathcal{O}^x_{\mathsf{write}}(i, j, d)$
if $\exists y \in \mathsf{CAP}, z.\ y = (i, z)$ then
  if $x$ then
    if $i \in Cor$ then
      $\langle \mathcal{A}(z, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle$
    else
      $r \leftarrow \langle \mathcal{C}_{\mathsf{write}}(z, j), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle$
      if $r \neq \mathsf{deny}$
        $\mathcal{DB} = r$
        $\mathcal{DB}' = \mathcal{DB}'[j \mapsto d]$
  else
    $\langle \mathcal{C}_{\mathsf{write}}(z, j, d), \mathcal{A}(\mathcal{DB})\rangle$
    if $\exists \mathcal{DB}'$ then
      $\mathcal{DB}' = \mathcal{DB}'[j \mapsto d]$
```

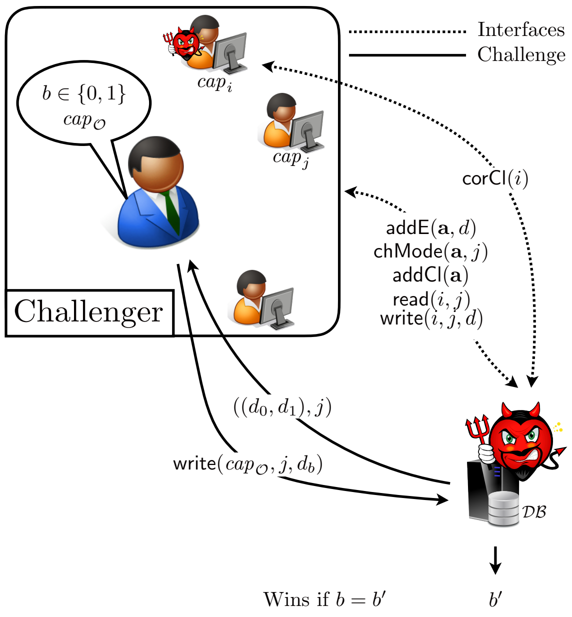
Figure 2.1.: Oracles accessible to the adversary in the game-based definitions.

We do that by defining a set of oracles, to a selection of which the adversary has access in every game. That selection is specified in each game separately. The oracles implement the respective functionality with which they are indexed. The flag x, which is present in the oracles $\mathcal{O}^x_{\mathsf{addE}}(\cdot)$, $\mathcal{O}^x_{\mathsf{chMode}}(\cdot)$, $\mathcal{O}^x_{\mathsf{read}}(\cdot)$, and $\mathcal{O}^x_{\mathsf{write}}(\cdot)$ determines where the database $\mathcal{DB}$ is stored. If the flag is set, $\mathcal{DB}$ is controlled by the challenger and the adversary only has access to it via oracles. If it is not set, the adversary controls the database. We use the sets CAP and Cor to keep track of issued capabilities and corrupted clients, respectively. Furthermore, in some cases we need an auxiliary database $\mathcal{DB}'$, which is supposed to maintain the correct state of the database, independent of the actions of the adversary. For instance, as we will later see in the definition of integrity, the adversary is supposed to inject a payload into the database $\mathcal{DB}$, which is not reflected in $\mathcal{DB}'$, which means that it is able to manipulate $\mathcal{DB}$ without holding access permissions. Finally, the oracle $\mathcal{O}_{\mathsf{query}}$ is required for the obliviousness definitions and contains a quite complex condition, which restricts the possible client and data indices that may be queried. This is necessary so as to rule out trivial attacks based on different access permissions for the challenge indices. The oracles are reported in Figure 2.1.

2.2.3.2. Secrecy

Intuitively, a Group ORAM preserves the secrecy of outsourced data if no party is able to deduce any information about the content of any entry she does not have access to. We formalize this intuition through a cryptographic game in the following definition, which is also illustrated below.

Definition 2.2 (Secrecy). *A Group ORAM* Π_{GORAM} preserves secrecy, *if* $\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, b) = 1\right]$ *is negligibly (in* λ*) close to* $1/2$ *for every* PPT *adversary* $\mathcal{A}$*, where* $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, b)$ *is the following game:*

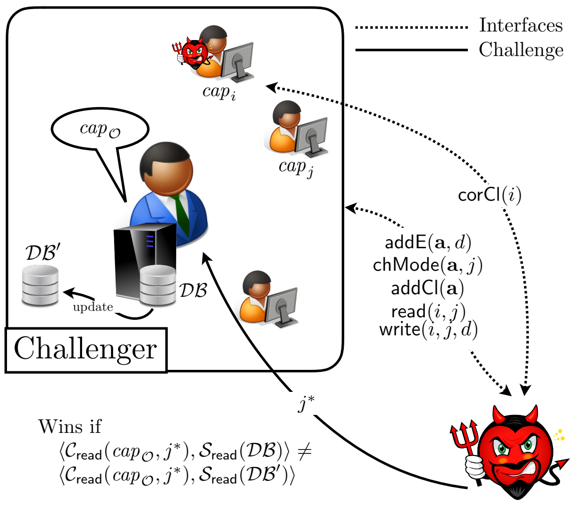


```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, b)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor) \leftarrow \mathsf{Setup}(1^\lambda, 0)$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{addCl}}(\cdot), \mathcal{O}^0_{\mathsf{addE}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{chMode}}(\cdot,\cdot), \mathcal{O}_{\mathsf{corCl}}(\cdot),$
    $\mathcal{O}^0_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
$(j, (d_0, d_1)) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathcal{DB})$
**if** $|d_0| = |d_1| \land \forall i \in Cor.\ \mathbf{AC}(i,j) = \bot$ **then**
  $\langle \mathcal{C}_{\mathsf{write}}(cap_{\mathcal{O}}, j, d_b), \mathcal{A}(\mathcal{DB}) \rangle$
  // $\mathbb{O}'$ *is defined as* $\mathbb{O}$ *with the difference that*
  //   $\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$ *aborts if* $\mathbf{a}(j) \neq \bot$,
  //   $\mathcal{O}_{\mathsf{corCl}}(i)$ *aborts if* $\mathbf{AC}(i,j) \neq \bot$, *and*
  //   $\mathcal{O}^0_{\mathsf{chMode}}(\mathbf{a}, j)$ *aborts if* $\exists i \in Cor.\ \mathbf{a}(i) \neq \bot$
  $b' \leftarrow \mathcal{A}^{\mathbb{O}'}()$
  **if** $b = b'$ **then**
    **return** 1
**return** 0
```

2.2.3.3. Integrity

A Group ORAM preserves the integrity of its entries if none of the clients can modify an entry to which she does not have write permissions. The respective cryptographic game is depicted next to the game formalization.

Definition 2.3 (Integrity). *A Group ORAM* Π_{GORAM} preserves integrity, *if* $\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{integrity}}(\lambda) = 1\right]$ *is negligible in* λ *for every* PPT *adversary* $\mathcal{A}$*, where* $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{integrity}}(\lambda)$ *is the following game:*

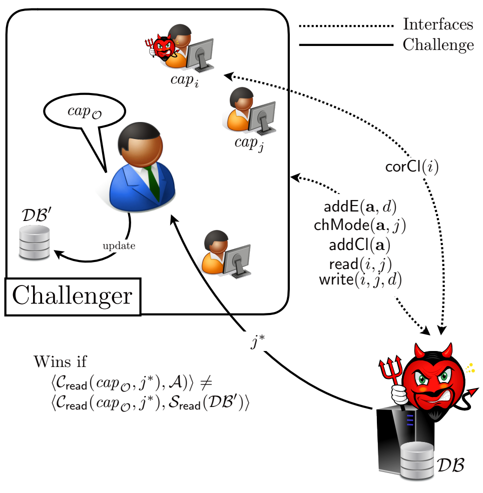

Interfaces
Challenge
cap_i
cap_j
$cap_{\mathcal{O}}$
$\mathcal{DB}'$
$\mathcal{DB}$
update
Challenger
$\mathsf{corCl}(i)$
$\mathsf{addE}(\mathbf{a}, d)$
$\mathsf{chMode}(\mathbf{a}, j)$
$\mathsf{addCl}(\mathbf{a})$
$\mathsf{read}(i, j)$
$\mathsf{write}(i, j, d)$
j^*
Wins if
$\langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle \neq$
$\langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}')\rangle$

```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{integrity}}(\lambda)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor, \mathcal{DB}') \leftarrow \mathsf{Setup}(1^\lambda, 1)$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{addCl}}(\cdot), \mathcal{O}^1_{\mathsf{addE}}(\cdot,\cdot), \mathcal{O}^1_{\mathsf{chMode}}(\cdot,\cdot), \mathcal{O}_{\mathsf{corCl}}(\cdot),$
    $\mathcal{O}^1_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}^1_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
$j^* \leftarrow \mathcal{A}^{\mathbb{O}}()$
**if** $\forall i \in Cor.\ \mathbf{AC}(i, j^*) \neq \mathsf{RW}$ **then**
    $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$
    **if** $d^* \neq \mathcal{DB}'(j^*)$ **then**
        **return** 1
**return** 0
```

2.2.3.4. Tamper-resistance

Intuitively, a Group ORAM is tamper-resistant if the server, even colluding with a subset of malicious clients, is not able to convince an honest client about the integrity of some maliciously modified data. Notice that this property refers to a strong adversarial model, where the adversary may arbitrarily misbehave and collude with clients. Naturally, tamper-resistance holds true only for entries which none of the corrupted clients had ever access to. The respective cryptographic game is depicted below next to its formalization.

Definition 2.4 (Tamper resistance). *A Group ORAM* Π_{GORAM} *is* tamper-resistant, *if for all* PPT *adversaries* $\mathcal{A}$*,* $\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1\right]$ *is negligible in* λ *where* $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda)$ *is the following game:*

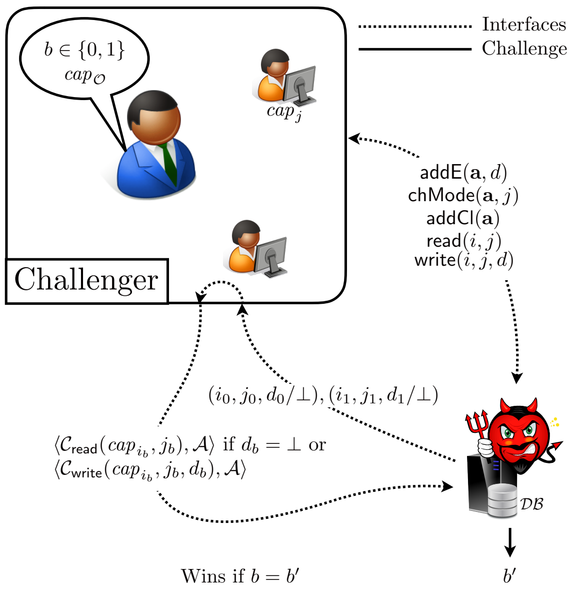

Interfaces
Challenge
cap_i
cap_j
$cap_{\mathcal{O}}$
$\mathcal{DB}'$
update
Challenger
$\mathsf{corCl}(i)$
$\mathsf{addE}(\mathbf{a}, d)$
$\mathsf{chMode}(\mathbf{a}, j)$
$\mathsf{addCl}(\mathbf{a})$
$\mathsf{read}(i, j)$
$\mathsf{write}(i, j, d)$
j^*
Wins if $\langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{A} \rangle \neq \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}') \rangle$
$\mathcal{DB}$

```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor, \mathcal{DB}') \leftarrow \mathsf{Setup}(1^\lambda, 1)$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{addCl}}(\cdot), \mathcal{O}^0_{\mathsf{addE}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{chMode}}(\cdot,\cdot), \mathcal{O}_{\mathsf{corCl}}(\cdot),$
    $\mathcal{O}^0_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
// Assume that $\mathbf{AC}(i,j)$ stores a list of permissions
//   accounting for the history of permissions
//   for client $i$ on index $j$
$j^* \leftarrow \mathcal{A}^{\mathbb{O}}(\mathcal{DB})$
if $\forall i, p.\ i \in Cor \wedge p \in \mathbf{AC}(i, j^*) \implies p = \bot$ then
  $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{A}(\mathcal{DB}) \rangle$
  if $d^* \neq \mathcal{DB}'(j^*)$ then
    return 1
return 0
```

2.2.3.5. Obliviousness

Intuitively, a Group ORAM is oblivious if the server cannot distinguish between two arbitrary query sequences which contain read and write operations. The cryptographic game is defined below and illustrated right next to the game formalization. We notice that the universally quantified restrictions in the definition of $\mathcal{O}_{\mathsf{query}}$ in Figure 2.1 concerning the corrupted clients have no effect on the if clause. The reason is that the set Cor is always empty due to the missing $\mathcal{O}_{\mathsf{corCl}}$ oracle. Hence, any universal quantification over Cor is true.

Definition 2.5 (Obliviousness). *A Group ORAM* Π_{GORAM} *is* oblivious, *if* $\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{obliv}}(\lambda, b) = 1\right]$ *is negligibly (in λ) close to $1/2$ for every* PPT *adversary $\mathcal{A}$, where* $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{obliv}}(\lambda, b)$ *is the following game:*

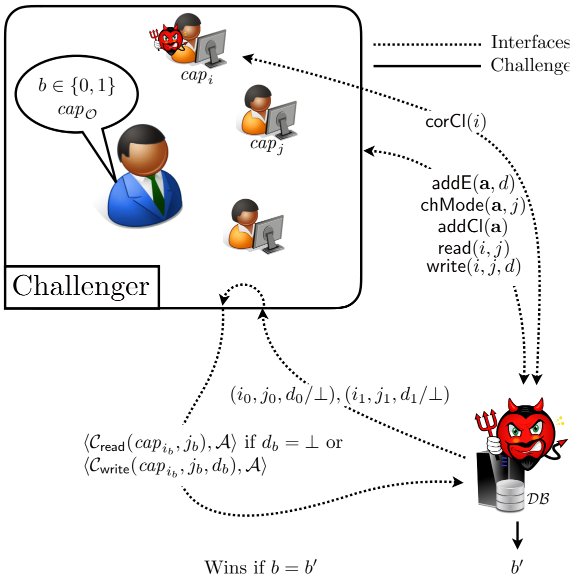

Interfaces
Challenge
$b \in \{0,1\}$ $cap_{\mathcal{O}}$
cap_j
Challenger
$\mathsf{addE}(\mathbf{a}, d)$
$\mathsf{chMode}(\mathbf{a}, j)$
$\mathsf{addCl}(\mathbf{a})$
$\mathsf{read}(i, j)$
$\mathsf{write}(i, j, d)$
$(i_0, j_0, d_0/\bot), (i_1, j_1, d_1/\bot)$
$\langle \mathcal{C}_{\mathsf{read}}(cap_{i_b}, j_b), \mathcal{A} \rangle$ if $d_b = \bot$ or $\langle \mathcal{C}_{\mathsf{write}}(cap_{i_b}, j_b, d_b), \mathcal{A} \rangle$
$\mathcal{DB}$
Wins if $b = b'$
b'

```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{obliv}}(\lambda, b)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor) \leftarrow \mathsf{Setup}(1^\lambda, 0)$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{addCl}}(\cdot), \mathcal{O}^0_{\mathsf{addE}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{chMode}}(\cdot,\cdot),$
    $\mathcal{O}_{\mathsf{query}}(b,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot), \mathcal{O}^0_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathcal{DB})$
if $b = b'$ then
  return 1
return 0
```

2.2.3.6. Obliviousness against malicious clients

Intuitively, a Group ORAM is oblivious against malicious clients if the server and an arbitrary subset of clients cannot get any information about the access patterns of honest clients, other than what is trivially leaked by the entries that the corrupted clients have read access to. The following definition, graphically supported by the picture next to the formal game description below, is an extension of the previous one which allows the adversary to corrupt arbitrary clients. However, in order to avoid trivial attacks, we restrict the queries of the adversary to the write oracle to indices that the set of corrupted clients cannot read. The query oracle already accounts for the further restriction since it aborts whenever it is called to write on indices to which any corrupted client has access to.

Definition 2.6 (Obliviousness against Malicious Clients). *A Group ORAM* Π_{GORAM} *is* oblivious against malicious clients, *if* $\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{m\text{-}obliv}}(\lambda, b) = 1\right]$ *is negligibly (in* λ*) close to* $1/2$ *for all* PPT *adversaries* $\mathcal{A}$*, where* $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{m\text{-}obliv}}(\lambda, b)$ *is the following game:*

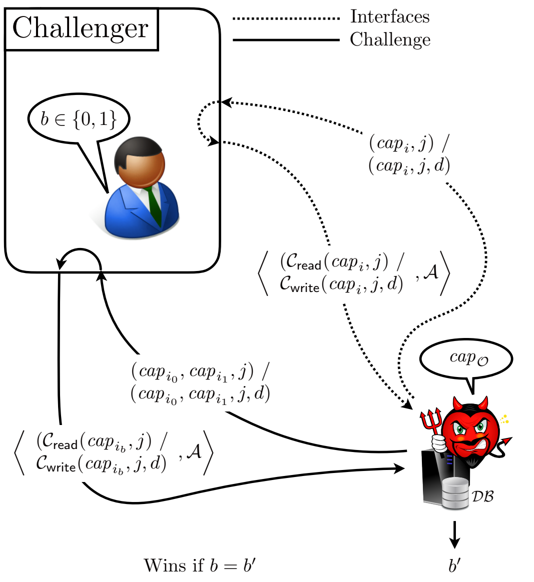

Interfaces
Challenge
$b \in \{0,1\}$ $cap_{\mathcal{O}}$
cap_i
cap_j
Challenger
corCl(i)
addE($\mathbf{a}, d$)
chMode($\mathbf{a}, j$)
addCl($\mathbf{a}$)
read(i, j)
write(i, j, d)
$(i_0, j_0, d_0/\bot), (i_1, j_1, d_1/\bot)$
$\langle \mathcal{C}_{\mathsf{read}}(cap_{i_b}, j_b), \mathcal{A}\rangle$ if $d_b = \bot$ or
$\langle \mathcal{C}_{\mathsf{write}}(cap_{i_b}, j_b, d_b), \mathcal{A}\rangle$
$\mathcal{DB}$
b'
Wins if $b = b'$

```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{m\text{-}obliv}}(\lambda, b)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor) \leftarrow \mathsf{Setup}(1^\lambda, 0)$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{addCl}}(\cdot), \mathcal{O}^0_{\mathsf{addE}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{chMode}}(\cdot,\cdot), \mathcal{O}_{\mathsf{corCl}}(\cdot),$
    $\mathcal{O}_{\mathsf{query}}(b,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot), \mathcal{O}^0_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
// If $\mathcal{O}_{\mathsf{query}}$ is called on $d_a \neq \bot$ for $a \in \{0,1\}$
// $\mathcal{O}_{\mathsf{chMode}}(i, j_a)$ aborts if $i \in Cor$ and
// $\mathcal{O}_{\mathsf{corCl}}(i)$ aborts if $\mathbf{AC}(i, j_a) \neq \bot$ from now on
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathcal{DB})$
if $b = b'$ then
    return 1
return 0
```

2.2.3.7. Anonymity

A Group ORAM preserves anonymity if the data owner cannot efficiently link a given operation to a client, among the set of clients having access to the queried index. We formalize anonymity as a cryptographic game in the following definition, accompanied by a graphical illustration.

Definition 2.7 (Anonymity). *A Group ORAM* Π_{GORAM} *preserves* anonymity, *if* $\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{anonymity}}(\lambda, b) = 1\right]$ *is negligibly (in* λ*) close to* $1/2$ *for every* PPT *adversary* $\mathcal{A}$*, where* $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{anonymity}}(\lambda, b)$ *is the following game:*

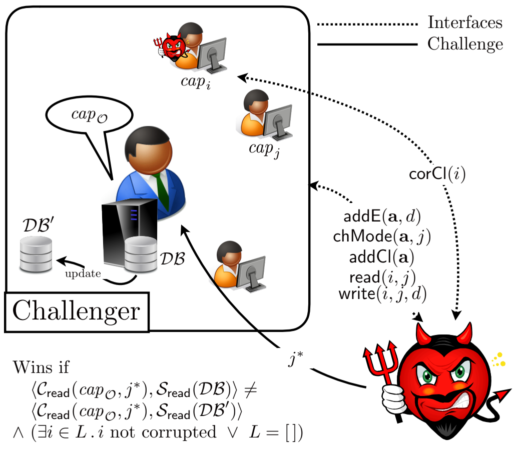

Challenger
Interfaces
Challenge
$b \in \{0,1\}$
(cap_i, j) /
(cap_i, j, d)
$\left\langle \begin{matrix}(\mathcal{C}_{\text{read}}(cap_i, j) \ / \\ \mathcal{C}_{\text{write}}(cap_i, j, d)\end{matrix}, \mathcal{A} \right\rangle$
$cap_{\mathcal{O}}$
$(cap_{i_0}, cap_{i_1}, j)$ /
$(cap_{i_0}, cap_{i_1}, j, d)$
$\left\langle \begin{matrix}(\mathcal{C}_{\text{read}}(cap_{i_b}, j) \ / \\ \mathcal{C}_{\text{write}}(cap_{i_b}, j, d)\end{matrix}, \mathcal{A} \right\rangle$
$\mathcal{DB}$
b'
Wins if $b = b'$

```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{anonymity}}(\lambda, b)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor) \leftarrow \mathsf{Setup}(1^\lambda, 0)$
  $\mathcal{O}_{\mathsf{read}}(cap_i, j)$                      $\mathcal{O}_{\mathsf{write}}(cap_i, j, d)$
  $\langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{A}(\mathcal{DB}) \rangle$      $\langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{A}(\mathcal{DB}) \rangle$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
$((cap_{i_0}, cap_{i_1}), j, d) \leftarrow \mathcal{A}^{\mathbb{O}}(cap_{\mathcal{O}}, \mathcal{DB})$
if $\mathbf{AC}(i_0, j) = \mathbf{AC}(i_1, j)$ then
  if $d = \perp$ then
    $\langle \mathcal{C}_{\mathsf{read}}(cap_{i_b}, j), \mathcal{A}(\mathcal{DB}) \rangle$
  else
    $\langle \mathcal{C}_{\mathsf{write}}(cap_{i_b}, j, d), \mathcal{A}(\mathcal{DB}) \rangle$
  $b' \leftarrow \mathcal{A}$
  if $b = b'$ then
    return 1
return 0
```

2.2.3.8. Accountable integrity

Intuitively, a Group ORAM preserves accountable integrity, if every client who modifies an entry without holding write permissions on that entry will be detected by an honest client. Clients detected of misbehavior are blamed and there is evidence in form of audit logs that can be used to hold them accountable. The blaming of clients must be correct in a natural way: (1) honest parties are never blamed and (2) in case of misbehavior by some party, at least one dishonest party is blamed. The literature defines this notion of accountability as *fairness* (cf. 1) and *completeness* (cf. 2) [123]. We define the accountable integrity property through a cryptographic game, illustrated next to its formalization below.

Definition 2.8 (Accountable integrity). *A Group ORAM* Π_{GORAM} *achieves* accountable integrity, *if* $\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right]$ *is negligible in* λ *for every* PPT *adversary* $\mathcal{A}$ *the following probability where* $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$ *is the following game:*


Interfaces
Challenge
cap_i
cap_j
$cap_{\mathcal{O}}$
$\mathsf{corCl}(i)$
$\mathsf{addE}(\mathbf{a}, d)$
$\mathsf{chMode}(\mathbf{a}, j)$
$\mathsf{addCl}(\mathbf{a})$
$\mathsf{read}(i, j)$
$\mathsf{write}(i, j, d)$
$\mathcal{DB}'$
update
$\mathcal{DB}$
Challenger
j^*
Wins if
$\langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}) \rangle \neq$
$\langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}') \rangle$
$\wedge\ (\exists i \in L \,.\, i$ not corrupted $\vee\ L = [\,])$

```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor, \mathcal{DB}') \leftarrow \mathsf{Setup}(1^\lambda, 1)$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{addCl}}(\cdot), \mathcal{O}^1_{\mathsf{addE}}(\cdot,\cdot), \mathcal{O}^1_{\mathsf{chMode}}(\cdot,\cdot), \mathcal{O}_{\mathsf{corCl}}(\cdot),$
    $\mathcal{O}^1_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}^1_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
$j^* \leftarrow \mathcal{A}^{\mathbb{O}}()$
if $\forall i \in Cor.\ \mathbf{AC}(i, j^*) \neq \mathsf{RW}$ then
  $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}) \rangle$
  $L \leftarrow \mathsf{blame}(cap_{\mathcal{O}}, \mathsf{Log}, j^*)$
  if $d^* \neq \mathcal{DB}'(j^*) \wedge ((\exists i \in L.\ i \notin Cor) \vee L = \emptyset)$ then
    return 1
return 0
```

Property	Server	Collusion
Secrecy	malicious	✓
Integrity	HbC	✗
Accountable Integrity	HbC	✗
Tamper-resistance	malicious	✓
Obliviousness	malicious	✗
Obliviousness	malicious	✓
Anonymity	malicious	✓

Table 2.1.: Security and privacy properties together with their minimal assumptions.

2.2.3.9. Discussion

Table 2.1 summarizes the security and privacy properties presented in this section, along with the corresponding assumptions. The HbC assumption is in fact only needed for integrity, since the correctness of client operations is checked by the server, thus avoiding costly operations on the client side. We will see in Section 2.7 that the HbC assumption is still needed for the accountable integrity property, since the server maintains a log of accesses, which allows for blaming misbehaving parties. Secrecy, tamper-resistance, and anonymity hold true even if the server is malicious and colludes with clients. Furthermore, we treat two variants of obliviousness, which differ in the non-collusion assumption: in the first variant, it holds only against a malicious server that does not collude with clients while the second variant allows for collusion. The rest of this chapter is organized with respect to these two variants: we first investigate on the collusion-enabled variant in Section 2.3, Section 2.4, and Section 2.5. Then we focus on the non-collusion variant in Section 2.6, Section 2.7, and Section 2.8. The reason for this organization is motivated by the improvement in efficiency that we gain with each further construction.

2.3. Computational Lower Bound

In this section, we study how much computational effort is necessary to realize a Group ORAM where obliviousness should hold against a server colluding with malicious clients. Our result shows that *any* construction, regardless of the underlying computational assumptions, must access the entire memory (up to a constant factor) in every operation. Our lower bound can be seen as a generalization of the result on history independence of Roche *et al.* [168], in the sense that they consider a “catastrophic attack” where the complete state of the client is leaked to the adversary, whereas we allow only the corruption of a certain subset of clients. Note that, while the bound in [168] concerns the *communication* complexity, our result only bounds the *computation* complexity on the server side.

2.3.1. Formal Result

In the following we state a formal lower bound on the computational complexity of any ORAM secure against malicious clients. We prove the theorem in Section B.2. We denote by physical addresses of a database the memory addresses associated with each storage cell of the memory. Intuitively, the lower bound says that the server has to access each entry of the dataset for any read and write operation.

Theorem 2.1. *Let n be the number of entries in the database and Π_{GORAM} be a Group ORAM scheme. If Π_{GORAM} accesses on average $o(n)$ physical addresses for each read and write operation (over the random coins of the read or write operation, respectively), Π_{GORAM} is not oblivious against malicious clients (see Definition 2.6).*

2.3.2. Discussion

Given the lower bound established in the previous section, we know that any Group ORAM scheme that is oblivious against malicious clients *must* read and write a fixed constant fraction of the database on every access. However, the bound does not impose any restriction on the required communication bandwidth. In fact, it does not exclude constructions with sublinear communication complexity, where the server performs a significant amount of computation. In particular, the aforementioned lower bound calls for the deployment of private information retrieval (PIR) [53] technologies, which allow a client to read an entry from a database without the server learning which entry it was.

The problem of private database modification is harder. A naïve approach would be to let the client change each entry in the database $\mathcal{DB}$ upon every access, which is however too expensive. Homomorphic encryption might be a natural candidate to outsource the computation to the server and to reduce the required bandwidth: unfortunately, Ostrovsky and Skeith III [152] showed that no private database modification (or PIR writing) scheme with sublinear communication (in the worst case) can be implemented using algebraic cryptographic constructions, such as linearly homomorphic encryption schemes. This result does not apply to schemes based on fully-homomorphic encryption, which is however hardly usable in practice due to the high computation cost of the currently known schemes.

The following sections describe our approach to bypass these two lower bounds. First we show how to integrate non-algebraic techniques, specifically out-of-band communication among clients, so as to achieve sublinear *amortized* communication complexity (Section 2.4). Second, we show how to leverage a trusted proxy performing the access to the server on behalf of clients in order to reach a logarithmic overhead in communication and server-side computation, with constant client-slide computation (Section 2.5).

2.4. PIR-GORAM

In this section, we present PIR-GORAM, a Group ORAM scheme based on PIR. Our construction is inspired by Franz *et al.* [81], who proposed to augment the database with a stack of modified entries, which is periodically flushed into the database by the data owner.

In our construction, we let each client $\mathcal{C}_i$ maintain its own temporary stack of entries S_i that is stored on the server side in addition to the regular database $\mathcal{DB}$. These stacks contain recent changes to entries in $\mathcal{DB}$ and to entries in other clients' stacks, which are not yet propagated to $\mathcal{DB}$. In contrast to the approach by Franz *et al.* [81], clients themselves are responsible to flush their stack once it is filled (i.e., after $|\mathsf{S}_i|$ many operations), *without requiring any intervention of the data owner*. An *oblivious gossiping protocol*, which can be realized using standard techniques [69, 118], allows clients to find the most up-to-date entry in the database, thereby obtaining a sublinear communication bandwith even for write operations and thus bypassing the impossibility result by Ostrovsky and Skeith III [152].

More precisely, when operating on index j, the client performs a PIR read on $\mathcal{DB}$ and on all stacks S_i, which can easily be realized since all stacks are stored on the server. Thanks to the oblivious gossiping protocol, the client knows which index is the most current one. At this point, the client appends either a dummy entry (read) or a real entry (write) to its personal stack. If the stack is full, the client flushes it. Flushing means to apply all changes in the personal stack to the database. To be oblivious, the client has to ensure that *all* entries in $\mathcal{DB}$ change. Moreover, for guaranteeing correctness, the client has to ensure that it does not overwrite entries which are more recent than those in its stack.

After explaining how to achieve obliviousness, we also need to discuss how to realize access control and how to protect the clients against the server. Data secrecy (i.e., read access control) is obtained via public-key encryption. Tamper-resistance (i.e., a-posteriori detection of illegal changes) is achieved by letting each client sign the modified entry so that others can check that this entry was produced by a client with write access. Data integrity (i.e., write access control) is achieved by further letting each client prove to the server that it is eligible to write the entry. As previously mentioned, data integrity is stronger than tamper-resistance, but assumes an honest-but-curious server: a malicious server may collude with malicious clients and thus store arbitrary information without checking integrity proofs.

2.4.1. Cryptographic Preliminaries

PIR-GORAM relies on the standard cryptographic primitives public-key encryption, digital signatures, private information retrieval (PIR), and non-interactive zero-knowledge proofs of knowledge (NIZK). We summarize the notation in Table 2.2 while we postpone a detailed description to Appendix A. We require two different public-key cryptosystems: one that is IND-CPA secure [89], which also supports randomization and additively homomorphic operations, and one that is IND-CCA secure [25]. We index encryption and decryption keys with the respective acronym to make explicit which notion we use.

2.4.2. System Assumptions

Data structures. $\mathcal{DB}$ stores up to N entries of size B each, hence the maximum capacity of $\mathcal{DB}$ is BN. The number of clients with access to $\mathcal{DB}$ is at most M. We assume that the server $\mathcal{S}$ has a storage capacity of $O(BN + \sum_{i=1}^{k} B\,|\mathsf{S}_i|)$ for the database and the client

Primitive	**Notation**
Public-key encryption	Π_{PKE}
Key generation	$(ek^x, dk^x) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$ where $x \in \{\mathsf{CPA}, \mathsf{CCA}\}$
Encryption	$c \leftarrow \mathsf{E}(ek, m)$
Decryption	$m \leftarrow \mathsf{D}(dk, c)$
Randomization	$c' \leftarrow \mathsf{Rnd}(ek, c, r)$
Additive homomorphism	$\mathsf{D}(dk, \mathsf{E}(ek, m) \otimes \mathsf{E}(ek, n)) = m + n$, $\mathsf{D}(dk, \alpha \cdot \mathsf{E}(ek, m)) = \alpha m$
Digital signatures	Π_{DS}
Key generation	$(vk, sk) \leftarrow \mathsf{Gen}_{\mathsf{DS}}(1^\lambda)$
Signing	$\sigma \leftarrow \mathsf{sign}(sk, m)$
Verification	$\{\top, \bot\} \leftarrow \mathsf{vfy}(vk, \sigma, m)$
Private information retrieval	Π_{PIR}
Query generation	$q \leftarrow \mathsf{prepRead}(\mathsf{DB}, i)$
Query execution	$r \leftarrow \mathsf{execRead}(\mathsf{DB}, q)$
Response decoding	$d \leftarrow \mathsf{decodeResp}(r)$
NIZK	$P = PK\{(\vec{x}) :\ F(\vec{x}, \vec{y})\}$, $\vec{x}$ hidden by P, $\vec{y}$ revealed by P

Table 2.2.: Notation for cryptographic primitives.

stacks; each $\mathcal{C}_i$ has a storage capacity of $O(|\mathsf{S}_i|\, B + N)$ to store the personal stack and a partial position map posmap, which is used to find the most current version of each entry; finally, $\mathcal{O}$ has a storage capacity of $O(N + B)$ to store the full position map and the access control matrix. While the position map is in general $O(N)$, this is usually much less than the storage size of $O(NB)$ [29] and can also be decreased to $O(1)$ by storing it in recursive ORAMs [187]. The database $\mathcal{DB}$ is accompanied by a private access control matrix $\mathbf{AC}$ that lets $\mathcal{O}$ manage the per-client permissions for each entry in $\mathcal{DB}$. The possible access rights are R (read-only), RW (read-write), and $\bot$ (no access). Finally, we assume authenticated broadcast channels among clients so as to gossip position map updates using standard techniques[3] [69, 118].

Database layout. We represent the logical database $\mathcal{DB}$ as a list of entries $\mathsf{DB} = E_1, \ldots, E_N$ and a list of stacks $\mathsf{S}_1, \ldots, \mathsf{S}_M$, one stack for every client. Both the database and the stacks are stored on the server. A stack is an entry list $\mathsf{S}_i = E_{j+1}, \ldots, E_{j+|\mathsf{S}_i|}$ where $j = \sum_{k=1}^{i-1} |\mathsf{S}_k|$. We denote by $\mathsf{S}_i(\ell)$ the ℓ-th entry of S_i. We write $\mathsf{S}_1 || \ldots || \mathsf{S}_M$ to express the list of entries in all stacks. Similarly, we count from 1 to $\sum_{i=1}^{M} |\mathsf{S}_i|$ to index an entry in $\mathsf{S}_1 || \ldots || \mathsf{S}_M$.

Client capabilities. We assume that every client $\mathcal{C}_i$ holds a key pair $(ek_i^{\mathsf{CPA}}, dk_i^{\mathsf{CPA}})$ for a CPA-secure encryption scheme as well as a key pair $(ek_i^{\mathsf{CCA}}, dk_i^{\mathsf{CCA}})$ for a CCA-secure encryption scheme where $(ek_i^{\mathsf{CPA}}, ek_i^{\mathsf{CCA}})$ are publicly known and $(dk_i^{\mathsf{CPA}}, dk_i^{\mathsf{CCA}})$ are $\mathcal{C}_i$'s private keys. Moreover, each client stores a position map posmap and version numbers $(vrs, vrs_{\mathcal{O}})$ for every entry it holds permissions on. These version numbers are necessary

[3]Gossiping is necessary since we do not trust the server for consistency.

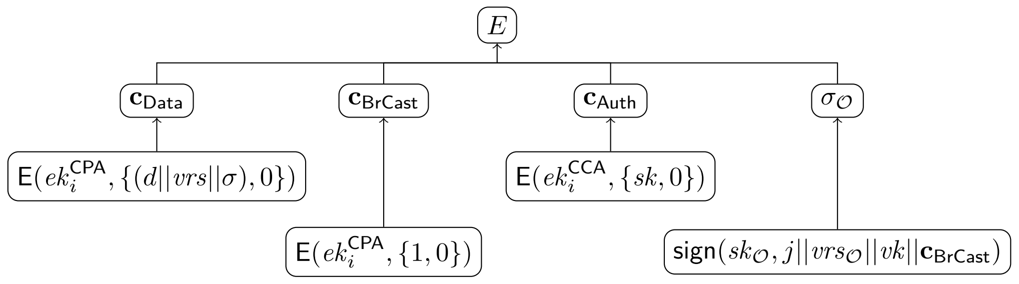

E
$\mathbf{c}_{\mathsf{Data}}$
$\mathbf{c}_{\mathsf{BrCast}}$
$\mathbf{c}_{\mathsf{Auth}}$
$\sigma_{\mathcal{O}}$
$\mathsf{E}(ek_i^{\mathsf{CPA}}, \{(d||vrs||\sigma), 0\})$
$\mathsf{E}(ek_i^{\mathsf{CCA}}, \{sk, 0\})$
$\mathsf{E}(ek_i^{\mathsf{CPA}}, \{1, 0\})$
$\mathsf{sign}(sk_{\mathcal{O}}, j||vrs_{\mathcal{O}}||vk||\mathbf{c}_{\mathsf{BrCast}})$

Figure 2.2.: The entry structure of an entry in the main database. If an entry resides on the stack, it contains only the part $\mathbf{c}_{\mathsf{Data}}$.

to prevent roll-back attacks (intuitively, the former on data, the latter on the access control matrix) and they are broadcast together with new indices for an entry upon write operations or whenever the policy is changed. Finally, every client stores a mapping which maps stack positions to entry indices: we keep this mapping implicit.

Entry Structure. An entry in the database DB has the form $E = (\mathbf{c}_{\mathsf{Data}}, \mathbf{c}_{\mathsf{BrCast}}, \mathbf{c}_{\mathsf{Auth}}, \sigma_{\mathcal{O}})$ where $\mathbf{c}_{\mathsf{Data}}$, $\mathbf{c}_{\mathsf{BrCast}}$, and $\mathbf{c}_{\mathsf{Auth}}$ are vectors of ciphertexts of length M and $\sigma_{\mathcal{O}}$ is a signature of the data owner $\mathcal{O}$. We describe each component and its functionality in the following (see also Figure 2.2).

$\underline{\mathbf{c}_{\mathsf{Data}}}$. The ciphertext $\mathbf{c}_{\mathsf{Data}}$ regulates read accesses. Specifically, it encrypts the data d of E for every client[4] with at least R permissions or a zero string for the others:

$$c^i_{\mathsf{Data}} = \begin{cases} \mathsf{E}(ek_i^{\mathsf{CPA}}, d||vrs||\sigma) & \text{if } \mathbf{AC}(i,j) \neq \bot \\ \mathsf{E}(ek_i^{\mathsf{CPA}}, 0^{|d|+|vrs|+|\sigma|}) & \text{otherwise} \end{cases} \tag{2.3}$$

where in addition to d, the ciphertext also contains the data version number vrs as well as a signature σ such that $\top = \mathsf{vfy}(vk, d||vrs)$. By vk we denote a verification key of a signature scheme which is different for every entry. An entry is valid if the verification of σ with vk outputs $\top$ and vk is the key authenticated by the data owner $\mathcal{O}$ in $\sigma_{\mathcal{O}}$, and invalid otherwise.

$\underline{\mathbf{c}_{\mathsf{BrCast}}}$. The ciphertext $\mathbf{c}_{\mathsf{BrCast}}$ is needed in the Broadcast protocol (described below), which is used to obliviously propagate a new entry index and new version numbers to other clients with read access. Specifically, it encrypts either 1 for every client with at least R permissions or zero for the others:

$$c^i_{\mathsf{BrCast}} = \begin{cases} \mathsf{E}(ek_i^{\mathsf{CPA}}, 1) & \text{if } \mathbf{AC}(i,j) \neq \bot \\ \mathsf{E}(ek_i^{\mathsf{CPA}}, 0) & \text{otherwise} \end{cases} \tag{2.4}$$

$\underline{\mathbf{c}_{\mathsf{Auth}}}$. This ciphertext contains the signing key corresponding to vk for those clients with

[4]This notation simplifies the presentation, but in the implementation we use of course hybrid encryption (cf. Section 2.10)

RW permissions or the zero string for the others. The exact form is

$$c^i_{\mathsf{Auth}} = \begin{cases} \mathsf{E}(ek_i^{\mathsf{CCA}}, sk) & \text{if } \mathbf{AC}(i,j) = \mathsf{RW} \\ \mathsf{E}(ek_i^{\mathsf{CCA}}, 0^{|sk|}) & \text{otherwise} \end{cases} \tag{2.5}$$

$\underline{\sigma_{\mathcal{O}}}$. The signature $\sigma_{\mathcal{O}}$ is created by $\mathcal{O}$ on the entry index j, a version number $vrs_{\mathcal{O}}$, the verification key vk, and $\mathbf{c}_{\mathsf{BrCast}}$.

Note that one cannot store the signing key sk in the entry $\mathbf{c}_{\mathsf{Data}}$. The reason is that whenever an entry is updated, the client needs to update all entries in the vector. However, for all entries except for its own, it does not know the private decryption key dk_i and thus, neither the corresponding private signing key nor the access rights for that entry. To update these entries, we exploit the homomorphic properties of the underlying encryption scheme, as explained below.

Update. Entries residing on a client's stack consist only of $\mathbf{c}_{\mathsf{Data}}$ in modified form where the old payload $D = d||vrs||\sigma$ has been replaced with $D' = d'||vrs'||\mathsf{sign}(sk, d'||vrs')$. Indeed, leveraging the homomorphic property and the structure of $\mathbf{c}_{\mathsf{BrCast}}$ (note that $\mathbf{c}_{\mathsf{BrCast}}$ is like $\mathbf{c}_{\mathsf{Data}}$, where D is replaced with 1) it is possible to generate $\mathbf{c}'_{\mathsf{Data}}$ as follows: choose r_i uniformly at random and compute

$$c^{i'}_{\mathsf{Data}} = \mathsf{Rnd}(ek_i^{\mathsf{CPA}}, c^i_{\mathsf{BrCast}} \cdot D', r_i). \tag{2.6}$$

Multiple data owners in one ORAM. The entry structure and database layout of PIR-GORAM can be easily extended in order to support multiple data owners storing their files in the same ORAM instance (think, e.g., of multiple patients storing their health record in the same ORAM), which is important to enhance user privacy (as the server does not even learn the owner of the accessed data). First, the signature $\sigma_{\mathcal{O}}$ is obviously constructed by the data owner to which the entry belongs. Most importantly, every entry might have a different set of potential readers and writers (e.g., not every patient visits the same doctors or pharmacies). As a consequence, an important invariant to maintain is that $\mathbf{c}_{\mathsf{Data}}$, $\mathbf{c}_{\mathsf{BrCast}}$, and $\mathbf{c}_{\mathsf{Auth}}$ are of equal length for every entry (i.e., the number of encryption keys used to construct them are the same), which can be easily achieved by padding. Otherwise, trivial entry-size based attacks against obliviousness are possible.

Obliviously broadcasting new indices. We propagate the updates of the entries to the clients with read access via broadcast. That is, if $E = (\mathbf{c}_{\mathsf{Data}}, \mathbf{c}_{\mathsf{BrCast}}, \mathbf{c}_{\mathsf{Auth}}, \sigma_{\mathcal{O}})$ with an old index j in the database DB and the new index ℓ on a stack or in DB, then we broadcast a message to all clients that can only be decrypted by clients having access to that entry. To this end, we leverage the same idea as in Equation (2.6), that is, we add the new index information to it. Clients compute $\mathbf{c}'_{\mathsf{BrCast}}$ with $c^{i'}_{\mathsf{BrCast}} = \mathsf{Rnd}(ek_i^{\mathsf{CPA}}, c^i_{\mathsf{BrCast}} \cdot (j||\ell||vrs'), r_i)$ for some random values r_i and a new version number vrs', and broadcast $\mathbf{c}'_{\mathsf{BrCast}}$ to all clients. We call this operation $\mathsf{Broadcast}((j||\ell||vrs'), \mathbf{c}_{\mathsf{BrCast}})$.

Clients having read access update their position map as follows. Upon receiving such a message c, the client $\mathcal{C}_i$ tries to decrypt the component corresponding to her identity with her private key dk_i^{CPA}. If the result is $(j||\ell||vrs')$ and not 0 (which means that it has

R access at least), then $\mathcal{C}_i$ updates its partial position map with the result. Otherwise it ignores the message. This protocol is oblivious since it is deterministically executed upon each operation and only clients with R access (which, as previously discussed, are excluded by the definition of obliviousness) can extract knowledge from the received ciphertext thanks to the CPA-security of the underlying encryption scheme.

Since malicious clients could potentially send wrong gossip messages about entries, e.g., claiming that an entry is residing in a different place than it actually is, we require that clients upload their broadcast messages also onto an append-only log, e.g., residing on the cloud, which is accessible by everyone. If a client does not find an entry using the latest index information, due to the malicious behavior of another client, then it just looks up the previous index and tries it there, and so on. Such append-only logs can be realized securely both in centralized [99] and decentralized [62] fashion. Utilizing such logs also enables accountability since the client who announced a wrong index is identifiable and, hence, blamable.

2.4.3. Algorithmic Description

Setup. The input to the setup algorithm is a list of data $d_1, \ldots, d_N$ and a list of clients $\mathcal{C}_1, \ldots, \mathcal{C}_M$ with an access control matrix $\mathbf{AC}$ which has an entry for every entry-client pair. The data owner first generates her own signing key pair $(vk_{\mathcal{O}}, sk_{\mathcal{O}}) \leftarrow \mathsf{Gen}_{\mathsf{DS}}(1^\lambda)$ and generates two encryption key pairs $(ek_j^{\mathsf{CPA}}, dk_j^{\mathsf{CPA}}) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}^{\mathsf{CPA}}(1^\lambda)$ and $(ek_j^{\mathsf{CCA}}, dk_j^{\mathsf{CCA}}) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}^{\mathsf{CCA}}(1^\lambda)$ for every client $\mathcal{C}_j$. Second, the data owner prepares every entry separately as follows: she generates a fresh signing key pair $(vk, sk) \leftarrow \mathsf{Gen}_{\mathsf{DS}}(1^\lambda)$ and sets up $\mathbf{c}_{\mathsf{Data}}$ as in Equation (2.3) using d_j and a version number 0, attaching a signature $\sigma = \mathsf{sign}(sk, d_j||0)$. $\mathbf{c}_{\mathsf{BrCast}}$ is generated as in Equation (2.4). Next, $\mathbf{c}_{\mathsf{Auth}}$ is generated as in Equation (2.5) using the just generated sk. Finally, using a data owner version number 0, $\mathcal{O}$ attaches $\sigma_{\mathcal{O}} = \mathsf{sign}(sk_{\mathcal{O}}, j||0||vk||\mathbf{c}_{\mathsf{BrCast}})$. $\mathcal{O}$ uploads all entries to $\mathcal{S}$ and broadcasts the client capabilities $cap_i = (\mathsf{posmap}_i, \vec{ek}, vk_{\mathcal{O}}, i_{\mathsf{S}}, len_{\mathsf{S}}, dk_i^{\mathsf{CPA}}, dk_i^{\mathsf{CCA}})$ where posmap_i is the full position map posmap restricted to those entries on which $\mathcal{C}_i$ holds at least R permissions, $\vec{ek}$ is a list of all clients' encryption keys, $i_{\mathsf{S}} = 0$ is $\mathcal{C}_i$'s current stack pointer, and len_{S} is the corresponding stack length. Notice that initially, posmap is the identity mapping on the domain $\{1, \ldots, N\}$ since all entries reside in the main database and the stacks are empty.

Reading and writing. To read or write to the database, clients have to perform two steps: extracting the data (Algorithm 1) and appending an entry to the personal stack (Algorithm 2 for writing and Algorithm 3 for reading).

To extract the payload, the client performs two PIR queries: one on DB for the desired index j and one on the concatenation of all stacks for either a more current version of j or an arbitrary one (lines 1.1–1.8): this is crucial to hide from the server the information on whether or not the client is retrieving a previously modified entry. It then checks the entry's authenticity as provided by $\sigma_{\mathcal{O}}$ and retrieves the verification key used for further verification (line 1.9). The client extracts henceforth the overall payload (line 1.11) from the most current entry (either in DB or on a stack (line 1.10)) and verifies its validity

Algorithm 1 $\{d, \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{extData}}(cap_i, j), \mathcal{S}_{\mathsf{extData}}(\mathcal{DB})\rangle$

```
Input: the client capability $cap_i$ and the desired index $j$ to extract the data from
Output: the data $d$ stored at $j$ or deny in case of failure
 1: $(\mathsf{posmap}, \vec{ek}, vk_{\mathcal{O}}, i_{\mathsf{S}}, len_{\mathsf{S}}, dk_i^{\mathsf{CPA}}, dk_i^{\mathsf{CCA}}) \leftarrow cap_i$
 2: $j' \leftarrow \mathsf{posmap}(j) - N$ if $\mathsf{posmap}(j) > N$, otherwise choose $j'$ uniformly at random from
    $\{1, \ldots, \sum_{i=1}^{M} |\mathsf{S}_i|\}$
 3: $q \leftarrow \mathsf{prepRead}(\mathsf{DB}, j)$
 4: $q' \leftarrow \mathsf{prepRead}(\mathsf{S}_1 || \ldots || \mathsf{S}_m, j')$
 5: Send $q, q'$ to $\mathcal{S}$
 6: Receive $r, r'$ from $\mathcal{S}$
 7: $E \leftarrow \mathsf{decodeResp}(r)$, $E' \leftarrow \mathsf{decodeResp}(r')$
 8: Parse $E$ as $(\mathbf{c}_{\mathsf{Data}}, \mathbf{c}_{\mathsf{BrCast}}, \mathbf{c}_{\mathsf{Auth}}, \sigma_{\mathcal{O}})$ and $E'$ as $(\mathbf{c}'_{\mathsf{Data}})$
 9: Abort if $\bot = \mathsf{vfy}(vk_{\mathcal{O}}, \sigma_{\mathcal{O}}, (j || vrs_{\mathcal{O}} || vk || \mathbf{c}_{\mathsf{BrCast}}))$ or $vrs_{\mathcal{O}}$ is not current
10: $\overline{\mathbf{c}_{\mathsf{Data}}} \leftarrow \mathbf{c}_{\mathsf{Data}}$ if $\mathsf{posmap}(j) \leq N$ and $\mathbf{c}'_{\mathsf{Data}}$ otherwise
11: $(d || vrs || \sigma) \leftarrow \mathsf{D}(dk_i^{\mathsf{CPA}}, \overline{c_{\mathsf{Data}}}^i)$
12: Abort if $\bot = \mathsf{vfy}(vk, \sigma, d || vrs)$ or $vrs$ is not current
13: if $i_{\mathsf{S}} > len_{\mathsf{S}}$ then
14:     $\mathsf{flush}(cap_i)$, $i_{\mathsf{S}} = 1$
15: end if
16: Increment $i_{\mathsf{S}}$
17: return $d$ if $d \neq \bot$ and deny otherwise
```

Algorithm 2 $\langle \mathcal{C}_{\mathsf{repl}}(cap_i, j, vrs, d', \mathbf{c}_{\mathsf{BrCast}}, \mathbf{c}_{\mathsf{Auth}}), \mathcal{S}_{\mathsf{repl}}(\mathcal{DB})\rangle$

```
Input: the client capability $cap_i$, the desired index $j$ to operate on, the old version
    number $vrs$ of the $j$-th entry, the new data $d'$, the broadcast ciphertext $\mathbf{c}_{\mathsf{BrCast}}$, and
    the authorization ciphertext $\mathbf{c}_{\mathsf{Auth}}$ of the $j$-th entry
 1: $(\mathsf{posmap}, \vec{ek}, vk_{\mathcal{O}}, i_{\mathsf{S}}, len_{\mathsf{S}}, dk_i^{\mathsf{CPA}}, dk_i^{\mathsf{CCA}}) \leftarrow cap_i$
 2: $sk \leftarrow \mathsf{D}(dk_i^{\mathsf{CCA}}, c_{\mathsf{Auth}}^i)$
 3: Increment $vrs$, $\sigma \leftarrow \mathsf{sign}(sk, d' || vrs)$
 4: Select $\vec{r}$ uniformly at random
 5: $c_{\mathsf{Data}}^{i'} \leftarrow \mathsf{Rnd}(ek_i^{\mathsf{CPA}}, c_{\mathsf{BrCast}}^i \cdot (d' || vrs || \sigma), r_i)$ for $i \in \{1, \ldots, M\}$
 6: Send $(i, i_{\mathsf{S}}, \mathbf{c}'_{\mathsf{Data}})$ to $\mathcal{S}$
 7: $\mathsf{Broadcast}((j || (N + \sum_{k=0}^{i-1} |\mathsf{S}_k| + i_{\mathsf{S}}) || vrs), \mathbf{c}_{\mathsf{BrCast}})$
```

(line 1.12). Before returning the extracted data (line 1.17), the client flushes the personal stack if it is full (lines 1.13–1.16). We explain this algorithm in the next paragraph. We stress that data extraction is performed independently of whether the client reads or writes. Note that up to this point, since the server only sees PIR queries, it cannot distinguish read and write.

The next step (i.e., adding an entry to the stack), however, requires more care in order to retain obliviousness. In particular, when writing, the client appends an entry to its personal stack that replaces the actual entry in DB (see Algorithm 2). In order to

Algorithm 3 $\langle \mathcal{C}_{\mathsf{addDummy}}(cap_i, \mathbf{c}_{\mathsf{BrCast}}), \mathcal{S}_{\mathsf{addDummy}}(\mathcal{DB})\rangle$

```
Input: the client capability $cap_i$ and the broadcast ciphertext $\mathbf{c}_{\mathsf{BrCast}}$
1: $(\mathsf{posmap}, \vec{ek}, vk_{\mathcal{O}}, i_{\mathsf{S}}, len_{\mathsf{S}}, dk_i^{\mathsf{CPA}}, dk_i^{\mathsf{CCA}}) \leftarrow cap_i$
2: Uniformly sample a vector $\mathbf{c}'_{\mathsf{Data}}$ of encryptions of 0
3: Send $(i, i_{\mathsf{S}}, \mathbf{c}'_{\mathsf{Data}})$ to $\mathcal{S}$
4: $\mathsf{Broadcast}(0, \mathbf{c}_{\mathsf{BrCast}})$
```

make read and write indistinguishable, when reading, the client appends an entry to its stack which is indistinguishable from a real entry since it is an entry on which no-one holds any permissions (see Algorithm 3). Finally, the client broadcasts the modified index information in write or a zero string in read.

Flushing the stack (Algorithm 4). The flush algorithm pushes the elements in the stack that are up-to-date to DB[5]. In particular, the client first builds an index structure that contains all elements that are up-to-date (ϕ, lines 4.2–4.9) based on the mapping of stack indices to real indices that the client stores implicitly. The client then downloads the stack (line 4.10) and changes every entry of DB (PIR writing). To this end, it downloads and uploads every entry $E_j \in \mathsf{DB}$ (lines 4.12, 4.21, and 4.27).

If the currently downloaded entry is outdated, the client takes the locally stored data from S_i and rerandomizes it (lines 4.14–4.18). Then it computes an *integrity proof* (technically, a NIZK) P that shows the following OR statement: either it is eligible to write the entry by proving that it knows the signing key (line 4.19) corresponding to the verification key (line 4.13) which is authenticated by the data owner, or it only rerandomized the data part (line 4.20). In that notation, the underscore _ refers to hidden variables in the proof that the client does not know.

In case there is no entry in the stack that is more recent, it rerandomizes the current entry in DB (line 4.25) and creates an integrity proof with the same statement as in the previous case, just that now the second part of the disjunction is true (line 4.26). In any case, the client broadcasts the new indices of all updated entries to all clients (line 4.22 for a real update and line 4.28 for a dummy update). We stress that the two proofs created in lines 4.20 and 4.26 are indistinguishable by the zero-knowledge property and hence do not reveal to the server whether the entry is updated or left unchanged, which is crucial for achieving the obliviousness of data accesses.

Adding new clients. To grant a new client $\mathcal{C}_i$ access to entries in $\mathcal{DB}$, $\mathcal{O}$ prepares a client capability cap_i as described above in the setup phase. In general, if not all capabilities are created initially, every entry has to be adapted when adding a new client as well as every client's capability. More precisely, for each entry, $\mathcal{O}$ adds a ciphertext to $\mathbf{c}_{\mathsf{Data}}$, $\mathbf{c}_{\mathsf{BrCast}}$ and $\mathbf{c}_{\mathsf{Auth}}$ and every client needs to learn ek_i^{CPA} and ek_i^{CCA}.

Adding new entries. To add a new entry to the database, $\mathcal{O}$ prepares it according to the entry structure and sends it to $\mathcal{S}$. Finally, $\mathcal{O}$ obliviously broadcasts the corresponding index information to all clients.

[5]Some elements may be outdated, since a different user may have the most recent version in its stack.

Algorithm 4 $\langle \mathcal{C}_{\mathsf{flush}}(cap_i), \mathcal{S}_{\mathsf{flush}}(\mathcal{DB})\rangle$

```
Input: the client capability $cap_i$
 1: $(\mathsf{posmap}, \vec{ek}, vk_{\mathcal{O}}, i_{\mathsf{S}}, len_{\mathsf{S}}, dk_i^{\mathsf{CPA}}, dk_i^{\mathsf{CCA}}) \leftarrow cap_i$
 2: Initialize $\phi = [\,]$
 3: $off \leftarrow |\mathsf{DB}| + \sum_{l=1}^{i-1} |\mathsf{S}_l|$
 4: for $\ell = len_{\mathsf{S}}$ down to 1 do
 5:     $j \leftarrow$ index of $\mathsf{S}_i(\ell)$ in $\mathsf{DB}$
 6:     if $\phi(j) = \bot$ and $\mathsf{posmap}(j) = \ell + off$ then
 7:         $\phi \leftarrow \phi[j \mapsto \ell]$, $\mathsf{posmap} \leftarrow \mathsf{posmap}[j \mapsto j]$
 8:     end if
 9: end for
10: Download $\mathsf{S}_i$ from $\mathcal{S}$
11: for $j = 1$ to $|\mathsf{DB}|$ do
12:     Download $E_j = (\mathbf{c}_{\mathsf{Data}}, \mathbf{c}_{\mathsf{BrCast}}, \mathbf{c}_{\mathsf{Auth}}, \sigma_{\mathcal{O}})$ from $\mathcal{S}$
13:     Extract $vk$ from $\sigma_{\mathcal{O}}$
14:     $l \leftarrow \phi(j)$
15:     if $l \neq \bot$ then
16:         Parse $\mathsf{S}_i(l)$ as $\mathbf{c}'_{\mathsf{Data}}$
17:         Select $\vec{r}$ uniformly at random
18:         $c^{i*}_{\mathsf{Data}} \leftarrow \mathsf{Rnd}(ek_i^{\mathsf{CPA}}, c^{i'}_{\mathsf{Data}}, r_i)$ for $1 \leq i \leq M$
19:         $sk \leftarrow \mathsf{D}(dk_i^{\mathsf{CCA}}, c^i_{\mathsf{Auth}})$
20:         $P \leftarrow PK \left\{ \begin{array}{l} (sk, \_) : \\ \quad (vk, sk) \text{ valid key pair } \vee \\ \quad \forall i.\ c^{i*}_{\mathsf{Data}} = \mathsf{Rnd}(ek_i^{\mathsf{CPA}}, c^i_{\mathsf{Data}}, \_) \end{array} \right\}$
21:         Send $P, E'_j = (\mathbf{c}^*_{\mathsf{Data}}, \mathbf{c}_{\mathsf{BrCast}}, \mathbf{c}_{\mathsf{Auth}}, \sigma_{\mathcal{O}})$ to $\mathcal{S}$
22:         $\mathsf{Broadcast}((j||j||vrs), \mathbf{c}_{\mathsf{BrCast}})$
23:     else
24:         Select $\vec{r}$ uniformly at random
25:         $c^{i*}_{\mathsf{Data}} \leftarrow \mathsf{Rnd}(ek_i^{\mathsf{CPA}}, c^i_{\mathsf{Data}}, r_i)$ for $1 \leq i \leq M$
26:         $P \leftarrow PK \left\{ \begin{array}{l} (\_, \vec{r}) : \\ \quad (vk, \_) \text{ valid key pair } \vee \\ \quad \forall i.\ c^{i*}_{\mathsf{Data}} = \mathsf{Rnd}(ek_i^{\mathsf{CPA}}, c^{i'}_{\mathsf{Data}}, r_i) \end{array} \right\}$
27:         Send $P, E'_j = (\mathbf{c}^*_{\mathsf{Data}}, \mathbf{c}_{\mathsf{BrCast}}, \mathbf{c}_{\mathsf{Auth}}, \sigma_{\mathcal{O}})$ to $\mathcal{S}$
28:         $\mathsf{Broadcast}(0, \mathbf{c}_{\mathsf{BrCast}})$ for random $r$
29:     end if
30: end for
```

Changing access permissions. To change access permissions of a certain entry, $\mathcal{O}$ modifies $\mathbf{c}_{\mathsf{Data}}$, $\mathbf{c}_{\mathsf{BrCast}}$ and/or $\mathbf{c}_{\mathsf{Auth}}$ as well as $\sigma_{\mathcal{O}}$ (with a new version number $vrs_{\mathcal{O}}$) accordingly.

2.4.4. Complexity Analysis

We elaborate on the communication complexity of our solution. We assume that $|\mathsf{DB}| = N$, that there M clients, and we set the stack length $len_{\mathsf{S}} = \sqrt{N}$ for every client. The worst case for an operation, hence, happens every $\sqrt{N}$-th operation for a client $\mathcal{C}_i$, meaning that besides extracting the data from the database and adding an entry to the personal stack, $\mathcal{C}_i$ has also to flush the stack. We analyze the four algorithms independently: extracting data requires two PIR reads, one on DB and the other on the concatenation of all stacks. Thus, the overall cost is $\mathsf{C_{PIR}}(N) + \mathsf{C_{PIR}}(M\sqrt{N})$ where $\mathsf{C_{PIR}}$ is a function mapping input sizes to communication complexity; $\mathsf{C_{PIR}}$ is to be replaced with concrete numbers when instantiating Π_{PIR}. Adding an entry to the personal stack always requires to upload one entry, independently of whether this replacement is real or dummy.

Our flushing algorithm assumes that $\mathcal{C}_i$ holds $\sqrt{N}$ entries and then down-and-uploads every entry of DB. Thus, the overall complexity is $2N + \sqrt{N}$. A similar analysis shows that if the client holds only $O(1)$ many entries, then $\mathcal{C}_i$ down-and-uploads DB but additionally performs a PIR step for every downloaded entry in its own stack to retrieve a potential replacement, resulting in a complexity of $2N + N \cdot \mathsf{C_{PIR}}(\sqrt{N})$.

To conclude, the construction achieves a worst-case complexity of $O(\mathsf{C_{PIR}}(N) + \mathsf{C_{PIR}}(M\sqrt{N}) + N)$ and $O(\mathsf{C_{PIR}}(N) + \mathsf{C_{PIR}}(M\sqrt{N}) + N\mathsf{C_{PIR}}(\sqrt{N}))$ for $O(\sqrt{N})$ and $O(1)$ client-side memory, respectively. By amortizing the flush step over $\sqrt{N}$ many operations, we achieve an amortized complexity of $O(\mathsf{C_{PIR}}(N) + \mathsf{C_{PIR}}(M\sqrt{N}) + \sqrt{N})$ or $O(\mathsf{C_{PIR}}(N) + \mathsf{C_{PIR}}(M\sqrt{N}) + \sqrt{N}\mathsf{C_{PIR}}(\sqrt{N}))$, respectively. Since our construction is parametric over the specific PIR protocol, we can leverage the progress in this field: at present, the best $\mathsf{C_{PIR}}(N)$ is $O(\log\log(N))$ [72] and, hence, the amortized cost becomes $O(\log\log(M\sqrt{N}) + \sqrt{N})$ or $O(\log\log(M\sqrt{N}) + \sqrt{N}\log\log(N))$, respectively. Since, in most scenarios, $M\sqrt{N} < 2^{2^{N/2}}$, we get $O(\sqrt{N})$ and $O(\sqrt{N}\log\log(N))$.

2.4.5. Variations

We discuss some variations of our constructions that achieve different assumptions and properties.

Malicious server. The construction, as we presented it, achieves integrity against an honest-but-curious server. If the server is malicious though, we cannot rely on it to verify the integrity proofs: a way to overcome this problem could be to force the server into honest behavior by letting him prove the correctness of his actions (e.g., using SMPC [47, 87, 133, 195]). This is, unfortunately, prohibitively expensive. Furthermore, since the whole database is solely stored on the server, it is clearly impossible to guarantee that the read operation is always performed correctly (e.g., the server could just go offline and therefore effectively erasing all the entries in the database) and thus to achieve any meaningful notion of integrity. However, we can still allow clients to detect a-posteriori illegal data changes, a security property that we address as tamper resistance. For achieving this property, integrity proofs are useless and can be dropped.

Relaxed AC privacy for better amortization. The analysis above shows that we achieve a $\sqrt{N}$ amortization factor, which is partially due to the fact that clients have to replace every entry of DB even if they do not change it or do not even have the rights to change it for keeping the access structure secret from the server. Assume that $\mathcal{C}_i$ has RW access to K_i many entries. If we were fine with giving up the privacy of $\mathbf{AC}$ and restricting obliviousness to those entries which may actually be changed by $\mathcal{C}_i$, then it would be sufficient in the flush algorithm to only exchange those K_i entries instead of DB in its entirety. Adapting the analysis and only considering the case where clients have $\sqrt{N}$ storage space, we get a worst case communication complexity of $O(\mathsf{C}_{\mathsf{PIR}}(N) + K_i)$ and an amortized communication complexity of $O(\mathsf{C}_{\mathsf{PIR}}(N) + \sqrt{N}/K_i)$. This means that the amortization factor is constant whenever $K_i = O(\sqrt{N})$.

2.4.6. Discussion

The construction presented in this section leverages PIR for reading entries and an accumulated PIR writing technique to replace old entries with newer ones. Due to the nature of PIR, one advantage of the construction is its possibility to allow multiple clients to concurrently read from the database and to append single entries to their stacks. This is no longer possible when a client flushes her personal stack since the database is entirely updated, which might lead to inconsistent results when reading from the database. To overcome this drawback, we present a fully concurrent, maliciously secure Group ORAM in Section 2.5. Another drawback of the flush algorithm is the cost of the integrity (zero-knowledge) proofs. Since we have to use public-key encryption as the top-layer encryption scheme for every entry to allow for proving properties about the underlying plaintexts, the number of proofs to be computed, naïvely implemented, is proportional to the block size. Varying block sizes require us to split an entry into chunks and encrypt every chunk separately since the message space of public-key encryption is a constant amount of bits. The zero-knowledge proof has then to be computed on every of these encrypted chunks. Our later construction for obliviousness only against the server, called GORAM, suffers a very similar problem. Hence, to overcome this linear dependency, we present two new proof paradigms. We present them in the context of GORAM, where the impact of integrity proofs is more severe than in PIR-GORAM. The first paradigm decreases the linear amount of zero-knowledge proofs to an amount that only depends on the security parameter (Section 2.6.5.1) while the second paradigm decreases this still linear amount (though in a different, much smaller parameter) to *one* proof (Section 2.6.5.2).

2.5. TAO-GORAM

Driven by the goal of building an efficient and scaleable Group ORAM that achieves obliviousness against malicious users, we explore the usage of a trusted proxy mediating accesses between clients and the server, an approach advocated in recent parallel ORAM constructions [29, 170, 184]. In contrast to previous works, we are not only interested in parallel accesses, but also in handling access control and providing obliviousness against multiple, possibly malicious, clients.

TaoStore [170]. In a nutshell, trusted proxy-based ORAM constructions implement a single-client ORAM which is run by the trusted entity on behalf of clients, which connect to it with read and write requests in a parallel fashion. We leverage the state of the art, TaoStore [170], which implements a variant of a Path-ORAM [187] client on the proxy and allows for retrieving multiple paths from the server concurrently. More specifically, the proxy consists of the *processor* and the *sequencer*. The processor performs read and write requests to the untrusted server: this is the most complex part of TaoStore and we leave it untouched. The sequencer is triggered by client requests and forwards them to the processor which executes them in a concurrent fashion.

Our modifications. Since the proxy is trusted, it can enforce access control. In particular, we can change the sequencer so as to let it know the access control matrix and check for every client's read and write requests whether they are eligible or not. As already envisioned by Sahin *et al.* [170], the underlying ORAM construction can be further refined in order to make it secure against a malicious server, either by following the approach based on Merkle-trees proposed by Stefanov *et al.* [187] or by using authenticated encryption as suggested by Sahin *et al.* [170]. In the rest of the chapter, we call the system TAO-GORAM.

2.6. GORAM

In this section, we first show how to realize a Group ORAM using a novel combination of ORAM, predicate encryption, and zero-knowledge proofs (Section 2.6.2 and Section 2.6.3). Since even the usage of the most efficient zero-knowledge proof system still yields an inefficient construction, we introduce two new proof techniques for boosting proofs of shuffle correctness (Section 2.6.5.1) and instantiate our general framework with those primitives.

2.6.1. Cryptographic preliminaries

Our construction relies on predicate encryption, public-key encryption, and NIZKs. We summarize the missing notation for predicate encryption in Table 2.3. Notice that $m \leftarrow \mathsf{D_{PE}}(psk_f, c)$ only returns a valid message m if $c \leftarrow \mathsf{E_{PE}}(ppk, x, m)$ such that $f(x) = 1$. Further details are postponed to Appendix A.

2.6.2. System Assumptions

Data structures and database layout. The layout of the database $\mathcal{DB}$ follows the one proposed by Stefanov *et al.* [187]. To store N data entries, we use a binary tree T of depth $D = O(\log N)$, where each node stores a bucket of entries, say b entries per bucket. We denote a node at depth d and row index i by $T_{d,i}$. The depth at the root ρ is 0 and increases from top to bottom; the row index increases from left to right, starting at 0. We often refer to the root of the tree as ρ instead of $T_{0,0}$. Moreover, Path-ORAM [187] uses a so-called *stash* as local storage to save entries that would overflow the root bucket. We

Primitive	**Notation**
Predicate encryption	Π_{PE}
Setup	$(pmsk, ppk) \leftarrow \mathsf{Gen}_{\mathsf{PE}}(1^\lambda, n)$
Key generation	$psk_f \leftarrow \mathsf{K}_{\mathsf{PE}}(pmsk, f)$
Encryption	$c \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk, x, m)$
Decryption	$m \leftarrow \mathsf{D}_{\mathsf{PE}}(psk_f, c)$
Randomization	$c' \leftarrow \mathsf{R}_{\mathsf{PE}}(ppk, c, r)$
Multiplicative homomorphism	$mn \leftarrow \mathsf{D}_{\mathsf{PE}}(psk_f, \mathsf{E}_{\mathsf{PE}}(ppk, x, m) \otimes \mathsf{E}_{\mathsf{PE}}(ppk, x, n))$
	$\alpha m \leftarrow \mathsf{D}_{\mathsf{PE}}(psk_f, \alpha \cdot \mathsf{E}_{\mathsf{PE}}(ppk, x, m))$

Table 2.3.: Additional notation for cryptographic primitives.

assume the stash to be stored and shared on the server like every other node, but we leave it out for the algorithmic description. The stash can also be incorporated in the root node, which does not carry b but $b + s$ entries where s is the size of the stash. The extension of the algorithms is straightforward (only the number of downloaded entries changes) and does not affect their computational complexity. In addition to the database, there is an index structure posmap that maps entry indices i to leaf indices l_i. If an entry index i is mapped in posmap to l_i then the entry with index i can be found in some node on the path from the leaf l_i to the root ρ of the tree. Finally, to initialize the database we fill it with dummy elements.

We assume that each client has a local storage of $O(\log N)$. Notice that the leaf index mapping has size $O(N)$, but the local client storage can be decreased to $O(\log N)$ by applying a standard ORAM construction recursively to it, as proposed by Shi *et al.* [178]. Additionally, the data owner stores a second database $\mathcal{ADB}$ that contains the attributes x_w and x_r associated to every entry in $\mathcal{DB}$ as well as predicates f_i associated to the client identities $\mathcal{C}_i$. Intuitively, $\mathcal{ADB}$ implements the access control matrix $\mathbf{AC}$ used in Definition 2.1. Since also $\mathcal{ADB}$ has size $O(N)$, we use the same technique as the one employed for the index structure. We further assume that clients establish authenticated channels with the server. These channels may be anonymous (e.g., by using anonymity networks [71] and anonymous credentials for the login [17–19, 143]), but not necessarily.

Client capabilities. Every client $\mathcal{C}_i$ holds a capability cap_i containing three different cryptographic keys: a decryption key dk for a public-key encryption scheme that serves as the top layer encryption of an entry in the tree and two predicate encryption secret keys $psk_{f_i}^{\mathsf{Auth}}$ and $psk_{f_i}^{\mathsf{Data}}$ for predicate f_i that regulates the client's access permissions.

Structure of an entry and access control modes. Abstractly, database entries are tuples of the form $E = (c_1, c_2, c_3)$ where $c_1, \ldots, c_3$ are ciphertexts obtained using a public-key encryption scheme (see Figure 2.7). In particular, c_1 is the encryption of an index j identifying the j-th entry of the database; c_2 is the encryption of a predicate encryption ciphertext c_{Auth}^j, which regulates the write access to the payload stored at j using the attribute x_w; c_3 is the encryption of a ciphertext c_{Data}^j, which is in turn the predicate

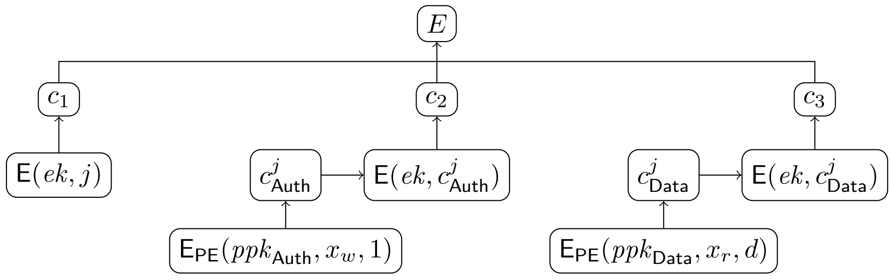

E
c_1
c_2
c_3
$\mathsf{E}(ek, j)$
c^j_{Auth}
$\mathsf{E}(ek, c^j_{\mathsf{Auth}})$
c^j_{Data}
$\mathsf{E}(ek, c^j_{\mathsf{Data}})$
$\mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_w, 1)$
$\mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Data}}, x_r, d)$

Figure 2.7.: The entry structure of an entry in the database.

Algorithm 5 $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda, n)$.

```
Input: security parameter $1^\lambda$, number of clients $n$
Output: the capability of the data owner $cap_{\mathcal{O}}$
1: $(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$
2: $(ppk_{\mathsf{Auth}}, pmsk_{\mathsf{Auth}}) \leftarrow \mathsf{Gen}_{\mathsf{PE}}(1^\lambda, n)$
3: $(ppk_{\mathsf{Data}}, pmsk_{\mathsf{Data}}) \leftarrow \mathsf{Gen}_{\mathsf{PE}}(1^\lambda, n)$
4: give $ek$ to the server $\mathcal{S}$
5: initialize $\mathcal{DB}$ on $\mathcal{S}$, $\mathcal{ADB} := \{\}$, $cnt_{\mathcal{C}} := 0$, $cnt_E := 0$
6: return $cap_{\mathcal{O}} := (cnt_{\mathcal{C}}, cnt_E, sk, pmsk_{\mathsf{Auth}}, pmsk_{\mathsf{Data}})$
```

encryption of the data d stored at position j.[6] We use the convention that an index $j > |\mathcal{DB}|$ indicates a dummy entry and we maintain the invariant that such entries are writable by each client.

Intuitively, using a client $\mathcal{C}_i$'s capability cap_i in order to implement the access control modes $\perp$, R, and RW on a data index j, we have to assign the attributes for an entry such that the following conditions hold: if $\mathcal{C}_i$'s mode for j is $\perp$, then $f_i(x_r) = f_i(x_w) = 0$, hence, $\mathcal{C}_i$ can neither decrypt c^j_{Auth} nor c^j_{Data}; if $\mathcal{C}_i$'s mode for j is R, then $f_i(x_w) = 0$ while $f_i(x_r) = 1$; finally, if $\mathcal{C}_i$'s mode for j is RW, then $f_i(x_w) = f_i(x_r) = 1$. Intuitively, in order to replace an entry, a client has to successfully prove that she can decrypt the ciphertext c^j_{Auth} and the result of that decryption is 1.

2.6.3. Algorithmic Description

Implementation of $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda, n)$ (Algorithm 5). Intuitively, the data owner initializes the cryptographic schemes (lines 5.1–5.3) as well as the rest of the infrastructure (lines 5.4–5.5), and finally outputs $\mathcal{O}$'s capability (line 5.6).[7] Notice that this algorithm takes as input the maximum number n of clients in the system, since this determines the size of the predicates ruling access control, which the predicate encryption schemes are

[6]Since encrypting a long payload using predicate encryption is expensive, the concrete instantiation that we evaluate in Section 2.10 uses hybrid encryption instead.

[7]For simplifying the notation, we assume for each encryption scheme that the public key is part of the secret key.

Algorithm 6 $\{cap_i, \mathsf{deny}\} \leftarrow \mathsf{addCl}(cap_{\mathcal{O}}, \mathbf{a})$.

Input: the capability of $\mathcal{O}$ $cap_{\mathcal{O}}$ and an access control list $\mathbf{a}$ for the client to be added
Output: a capability cap_i for client $\mathcal{C}_i$ in case of success, deny otherwise

```
1: $(cnt_{\mathcal{C}}, cnt_E, dk, pmsk_{\mathsf{Auth}}, pmsk_{\mathsf{Data}}) \leftarrow cap_{\mathcal{O}}$
2: if $|\mathbf{a}| \neq cnt_E$ then
3:     return deny
4: end if
5: $cnt_{\mathcal{C}} := cnt_{\mathcal{C}} + 1$
6: compute $f_i$ s.t. the following holds for $1 \leq j \leq |\mathbf{a}|$ and all $(x_{w,j}, x_{r,j}) := \mathcal{ADB}(j)$
       if $\mathbf{a}(j) = \perp$ then $f_i(x_{w,j}) = f_i(x_{r,j}) = 0$
       if $\mathbf{a}(j) = \mathsf{R}$ then $f_i(x_{w,j}) = 0$ and $f_i(x_{r,j}) = 1$
       if $\mathbf{a}(j) = \mathsf{RW}$ then $f_i(x_{w,j}) = f_i(x_{r,j}) = 1$
7: $\mathcal{ADB} := \mathcal{ADB}[\mathcal{C}_i \mapsto f_i]$
8: $psk_{f_i}^{\mathsf{Auth}} \leftarrow \mathsf{K}_{\mathsf{PE}}(pmsk_{\mathsf{Auth}}, f_i)$
9: $psk_{f_i}^{\mathsf{Data}} \leftarrow \mathsf{K}_{\mathsf{PE}}(pmsk_{\mathsf{Data}}, f_i)$
10: return $cap_i := (dk, psk_{f_i}^{\mathsf{Auth}}, psk_{f_i}^{\mathsf{Data}})$
```

parameterized by.

Implementation of $\{cap_i, \mathsf{deny}\} \leftarrow \mathsf{addCl}(cap_{\mathcal{O}}, \mathbf{a})$ **(Algorithm 6).** This algorithm allows $\mathcal{O}$ to register a new client in the system. Specifically, $\mathcal{O}$ creates a new capability for the new client $\mathcal{C}_i$ according to the given access permission list $\mathbf{a}$ (lines 6.6–6.10). If $\mathcal{O}$ wants to add more clients than n, the maximum number she initially decided, she can do so at the price of re-initializing the database. In particular, she has to setup new predicate encryption schemes, since these depend on n. Secondly, she has to distribute new capabilities to all clients. Finally, for each entry in the database, she has to re-encrypt the ciphertexts c_{Auth} and c_{Data} with the new keys.

Implementation of $\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ **(Algorithm 7).** In this algorithm, $\mathcal{O}$ adds a new entry that contains the payload d to the database. Furthermore, the new entry is protected according to the given access permission list $\mathbf{a}$. Intuitively, $\mathcal{O}$ assigns the new entry to a random leaf and downloads the corresponding path in the database (lines 7.5–7.6). It then creates the new entry and substitutes it for a dummy entry (lines 7.7–7.10). Finally, $\mathcal{O}$ rerandomizes the entries so as to hide from $\mathcal{S}$ which entry changes, and finally uploads the modified path to $\mathcal{S}$ (lines 7.11–7.15).

Eviction. In all ORAM constructions, the client has to rearrange the entries in the database in order to make subsequent accesses unlinkable to each other. In the tree construction we use [187], this is achieved by first assigning a new, randomly picked, leaf index to the read or written entry. After that, the entry might no longer reside on the path from the root to its designated leaf index and, thus, has to be moved. This procedure is called *eviction* (Algorithm 8).

This algorithm assigns the entry to be evicted to a new leaf index (line 8.1). It then locally shuffles and rerandomizes the given path according to a permutation π (lines

Algorithm 7 $\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$.

```
Input: the capability of $\mathcal{O}$ $cap_{\mathcal{O}}$, an access control list $\mathbf{a}$ and the data $d$ for the entry to be added
Output: a changed database $\mathcal{DB}'$ on $\mathcal{S}$ in case of success, deny otherwise
1: $(cnt_{\mathcal{C}}, cnt_E, dk, pmsk_{\mathsf{Auth}}, pmsk_{\mathsf{Data}}) \leftarrow cap_{\mathcal{O}}$
2: if $|\mathbf{a}| \neq cnt_{\mathcal{C}}$ then
3:     return deny
4: end if
5: $cnt_E := cnt_E + 1$, $j := cnt_E$, $l_j \leftarrow \{0,1\}^D$, $\mathsf{posmap} := \mathsf{posmap}[j \mapsto l_j]$
6: let $E_1, \ldots, E_{b(D+1)}$ be the path from $\rho$ to $T_{D,l_j}$ downloaded from $\mathcal{S}$ ($E_i = (c_{1,i}, c_{2,i}, c_{3,i})$)
7: let $k$ be such that $\mathsf{D}(dk, c_{1,k}) > |\mathcal{DB}|$
8: compute $(x_{w,j}, x_{r,j})$ s.t. the following holds for $1 \leq i \leq |\mathbf{a}|$ and all $f_i := \mathcal{ADB}(\mathcal{C}_i)$
       if $\mathbf{a}(i) = \perp$ then $f_i(x_{w,j}) = f_i(x_{r,j}) = 0$
       if $\mathbf{a}(i) = \mathsf{R}$ then $f_i(x_{w,j}) = 0$, $f_i(x_{r,j}) = 1$
       if $\mathbf{a}(i) = \mathsf{RW}$ then $f_i(x_{w,j}) = f_i(x_{r,j}) = 1$
9: $\mathcal{ADB} := \mathcal{ADB}[j \mapsto (x_{w,j}, x_{r,j})]$
10: $E_k := (c_{1,k}, c_{2,k}, c_{3,k})$ where
       $c^j_{\mathsf{Auth}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_{w,j}, 1)$
       $c^j_{\mathsf{Data}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Data}}, x_{r,j}, d)$
       $c_{1,k} \leftarrow \mathsf{E}(ek, j)$
       $c_{2,k} \leftarrow \mathsf{E}(ek, c^j_{\mathsf{Auth}})$
       $c_{3,k} \leftarrow \mathsf{E}(ek, c^j_{\mathsf{Data}})$
11: for all $1 \leq \ell \leq b(D+1)$, $\ell \neq k$ do
12:     select $r_\ell$ uniformly at random
13:     $E'_\ell \leftarrow \mathsf{Rnd}(pk, E_\ell, r_\ell)$
14: end for
15: upload $E'_1, \ldots, E'_{k-1}, E_k, E'_{k+1}, \ldots, E'_{b(D+1)}$ to $\mathcal{S}$
```

8.2–8.4). After replacing the old path with a new one, the evicted entry is supposed to be stored in a node along the path from the root to the assigned leaf, which always exists since the root is part of the permuted nodes. A peculiarity of our setting is that clients are not trusted and, in particular, they might store a sequence of ciphertexts in the database that is not a permutation of the original path (e.g., they could store a path of dummy entries, thereby cancelling the original data).

Integrity proofs. To tackle this problem, a first technical novelty in our construction is, in the read and write protocols, to let the client output the modified path along with a proof of shuffle correctness [22, 49], which has to be verified by the server ($s = 1$, lines 8.6–8.7). As the data owner is assumed to be honest, she does not have to send a proof in the chMode protocol ($s = 0$, line 8.9).

Implementation of $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ (Algorithm 9). In this protocol, $\mathcal{O}$ changes the access mode of the j-th entry in $\mathcal{DB}$ according to the new access permission list

Algorithm 8 $(E''_1, \dots, E''_{b(D+1)}, \pi, [P]) \leftarrow \mathsf{Evict}(E_1, \dots, E_{b(D+1)}, s, j, k)$.

```
Input: a list of entries $E_1, \dots, E_{b(D+1)}$, a bit $s$, an index $j$, and a position $k$ in the list
Output: a permuted and rerandomized list of entries $E''_1, \dots, E''_{b(D+1)}$, a permutation $\pi$,
    and a proof of shuffle correctness (if $s = 1$)
 1: $l_j \leftarrow \{0,1\}^D$, $\mathsf{posmap} := \mathsf{posmap}[j \mapsto l_j]$
 2: compute a permutation $\pi$ s.t. $\pi(k) = 1$ and for all other $\ell \neq k$, $\pi$ pushes $\ell$ down on the
    path from $\rho$ $(= E_1, \dots, E_b)$ to the current leaf node $(= E_{bD+1}, \dots, E_{b(D+1)})$ as long as
    the index of the $\ell$-th entry still lies on the path from $\rho$ to its designated leaf node.
 3: $E'_1, \dots, E'_{b(D+1)} := E_{\pi^{-1}(1)}, \dots, E_{\pi^{-1}(b(D+1))}$
 4: let $E''_1, \dots, E''_{b(D+1)}$ be the rerandomization of $E'_1, \dots, E'_{b(D+1)}$ as described in 7.11–7.14
    (including $k$)
 5: if $s = 1$ then
 6:     $P := PK\left\{ \begin{array}{l} (\pi, r_1, \dots, r_{b(D+1)}) : \\ \quad \forall \ell.\ E_\ell = \mathsf{Rnd}(ek, E_{\pi^{-1}(\ell)}, r_\ell) \end{array} \right\}$
 7:     return $E''_1, \dots, E''_{b(D+1)}, \pi, P$
 8: else
 9:     return $E''_1, \dots, E''_{b(D+1)}, \pi$
10: end if
```

a. Intuitively, she does so by downloading the path where the entry resides on (lines 9.5–9.6), changing the entry accordingly (lines 9.7–9.12), and uploading a modified and evicted path to the server (lines 9.13–9.14).

Implementation of $\{d, \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ (Algorithm 10). Intuitively, the client downloads the path which index j is assigned to and searches for the corresponding entry (lines 10.5–10.9). She then evicts the downloaded path, subject to the restriction that some dummy entry afterwards resides in the top position of the root node (lines 10.10–10.11). $\mathcal{C}$ uploads the evicted path together with a proof of shuffle correctness to $\mathcal{S}$ who verifies the proof and replaces the old with the new path in case of successful verification (line 10.12).

Obliviousness in presence of integrity proofs. $\mathcal{C}$ could in principle stop here since she has read the desired entry. However, in order to fulfill the notion of obliviousness (Definition 2.5), the read and write operations must be indistinguishable. In single-client ORAM constructions, $\mathcal{C}$ can make write indistinguishable from read by simply modifying the content of the desired entry before uploading the shuffled path to the server. This approach does not work in our setting, due to the presence of integrity proofs. Intuitively, in read, it would suffice to produce a proof of shuffle correctness, but this proof would not be the same as the one used in write, where one element in the path changes. Hence another technical novelty in our construction is the last part of the read protocol (lines 10.13–10.17), which "simulates" the write protocol despite the presence of integrity proofs. This is explained below, in the context of the write protocol.

Implementation of $\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle$ (Algorithm 11). Firstly,

Algorithm 9 $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$.

```
Input: the capability of $\mathcal{O}$ $cap_{\mathcal{O}}$, an access control list $\mathbf{a}$, and an index $j$
Output: deny if the algorithm fails
1: $(cnt_{\mathcal{C}}, cnt_E, dk, pmsk_{\mathsf{Auth}}, pmsk_{\mathsf{Data}}) \leftarrow cap_{\mathcal{O}}$
2: if $|\mathbf{a}| \neq cnt_{\mathcal{C}}$ or $j > cnt_E$ then
3:     return deny
4: end if
5: $l_j := \mathsf{posmap}(j)$
6: let $E_1, \ldots, E_{b(D+1)}$ be the path from $\rho$ to $T_{D,l_j}$ downloaded from $\mathcal{S}$ $(E_i = (c_{1,i}, c_{2,i}, c_{3,i}))$
7: let $k$ be s.t. $\mathsf{D}(dk, c_{1,k}) = j$
8: $(x'_{w,j}, x'_{r,j}) := \mathcal{ADB}(j)$
9: let $f$ be s.t. $f(x'_{r,j}) = 1$
10: $psk_f \leftarrow \mathsf{K}_{\mathsf{PE}}(pmsk_{\mathsf{Data}}, f)$
    $c^j_{\mathsf{Data}} \leftarrow \mathsf{D}(dk, c_{3,k})$
    $d \leftarrow \mathsf{D}_{\mathsf{PE}}(psk_f, c^j_{\mathsf{Data}})$
11: compute $(x_{w,j}, x_{r,j})$ according to 7.8 and subject to all $f_i$ in $\mathcal{ADB}$, also add them to
    $\mathcal{ADB}$ (7.9)
12: compute $E'_k$ as in 7.10
13: $(E''_1, \ldots, E''_{b(D+1)}, \pi) := \mathsf{Evict}(E_1, \ldots, E_{k-1}, E'_k, E_{k+1}, \ldots, E_{b(D+1)}, 0, j, k)$
14: upload $E''_1, \ldots, E''_{b(D+1)}$ to $\mathcal{S}$
```

$\mathcal{C}$ reads the element that she wishes to change (line 11.1). Secondly, $\mathcal{C}$ evicts the path with the difference that here the first entry in the root node is the element that $\mathcal{C}$ wants to change, as opposed to a dummy entry like in read (line 11.10). It is important to observe that the shuffle proof sent to the server (line 8.6) is indistinguishable in read and write since it hides both the permutation and the randomness used to rerandomize the entries. So far, we have shown how $\mathcal{C}$ can upload a shuffled and rerandomized path to the server without modifying the content of any entry.

In write, $\mathcal{C}$ can now replace the first entry in the root node with the entry containing the new payload (lines 11.12–11.13). In read, this step is simulated by rerandomizing the first entry of the root node, which is a dummy entry (line 10.15).

The integrity proofs P_{Auth} and P_{Ind} produced in read and write are indistinguishable (lines 10.14 and 10.16 for both): in both cases, they prove that $\mathcal{C}$ has the permission to write on the first entry of the root node and that the index has not changed. Notice that this proof can be produced also in read, since all clients have write access to dummy entries.

Permanent Entries. Some application scenarios of GORAM might require determined entries of the database not to be modifiable nor deletable, not even by the data owner herself (for instance, in the case of PHRs, the user should not be able to cancel diagnostic results in order to pay lower insurance fees). Even though we did not explicitly describe the construction, we mention that such a property can be achieved by assigning a binary attribute (*modifiable* or *permanent*) to each entry and storing a commitment to this in the

Algorithm 10 $\{d, \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$.

Input: the capability of the client executing the protocol cap_i and the index j to be read
Output: the data payload d in case of success, deny otherwise

```
 1: $(dk, psk_f^{\mathsf{Auth}}, psk_f^{\mathsf{Data}}) \leftarrow cap_i$
 2: **if** $j > |\mathcal{DB}|$ **then**
 3:     **return** $\mathsf{deny}$
 4: **end if**
 5: $l_j := \mathsf{posmap}(j)$
 6: let $E_1, \dots, E_{b(D+1)}$ and $k$ be as in lines 9.6–9.7
 7: Parse $E_k$ as $(c_{1,k}, c_{2,k}, c_{3,k})$
 8: $c^k_{\mathsf{Data}} \leftarrow \mathsf{D}(dk, c_{3,k})$
 9: $d \leftarrow \mathsf{D}_{\mathsf{PE}}(psk_f^{\mathsf{Data}}, c^k_{\mathsf{Data}})$
10: let $\ell$ be s.t. $\mathsf{D}(dk, c_{1,\ell}) > |\mathcal{DB}|$
11: $(E'_1, \dots, E'_{b(D+1)}, \pi, P) := \mathsf{Evict}(E_1, \dots, E_{b(D+1)}, 1, j, \ell)$
12: upload $E'_1, \dots, E'_{b(D+1)}$ and $P$ to $\mathcal{S}$
13: $c^j_{\mathsf{Auth}} \leftarrow \mathsf{D}(sk, c'_{2,1})$
14: $P_{\mathsf{Auth}} := PK\left\{\left(psk_f^{\mathsf{Auth}}\right) : \mathsf{D}_{\mathsf{PE}}(psk_f^{\mathsf{Auth}}, c^j_{\mathsf{Auth}}) = 1\right\}$
15: $E''_1 := (c''_{1,1}, c''_{2,1}, c''_{3,1})$ where
      $r_1, r_2, r_3$ are selected uniformly at random
      $c''_{l,1} \leftarrow \mathsf{Rnd}(ek, c'_{l,1}, r_l)$ for $l \in \{1, 3\}$
      $c''_{2,1} \leftarrow \mathsf{E}(ek, \mathsf{R}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, c^j_{\mathsf{Auth}}, r_2))$
16: $P_{\mathsf{Ind}} := PK\left\{(r_1) : c''_{1,1} = \mathsf{Rnd}(ek, c'_{1,1}, r_1)\right\}$
17: upload $E''_1$, $P_{\mathsf{Auth}}$, $P_{\mathsf{Ind}}$, and the necessary information to access $c^j_{\mathsf{Auth}}$ to $\mathcal{S}$
```

database. Every party that tries to modify a given entry, including the data owner, has to provide a proof that the respective attribute is set to *modifiable.* This can be efficiently instantiated using ElGamal encryption and Σ-protocols.

2.6.4. Complexity Analysis

The computational and communication complexity of our construction, for both the server and the client, is $O((B+G)\log N)$ where N is the number of the entries in the database, B is the block size of the entries in the database, and G is the number of clients that have access to the database. $O(B\log N)$ originates from the ORAM construction and we add $O(G\log N)$ for the access structure. Hence, our solution only adds a small overhead to the standard ORAM complexity. The client-side storage is $O(B\log N)$, while the server has to store $O(BN)$ many data.

Algorithm 11 $\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle$.

Input: the capability of the client executing the protocol cap_i, the index j to be written, and the data d

Output: deny if the algorithm fails

```
1:  execute 10.1–10.9, call the old payload $d'$
10: $(E'_1, \dots, E'_{b(D+1)}, \pi, P)$:=$\mathsf{Evict}(E_1, \dots, E_{b(D+1)}, 1, j, k)$
11: execute 10.12–10.14
12: $E''_1 := (c''_{1,1}, c''_{2,1}, c''_{3,1})$ where
        $r_1, r_2, r_3$ are selected uniformly at random
        $c''_{1,1} \leftarrow \mathsf{Rnd}(ek, c'_{1,1}, r_1)$
        $c''_{2,1} \leftarrow \mathsf{E}(ek, \mathsf{R}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, c^j_{\mathsf{Auth}}, r_2))$
        $c''_{3,1} \leftarrow \mathsf{E}(ek, \mathsf{R}_{\mathsf{PE}}(ppk_{\mathsf{Data}}, c^j_{\mathsf{Data}} \cdot d'^{-1} \cdot d, r_3))$
13: execute 10.16–10.17
```

2.6.5. Integrity Proofs Revisited

A zero-knowledge proof of shuffle correctness of a set of ciphertexts proves in zero-knowledge that a new set of ciphertexts contains the same plaintexts in permuted order. In our system the encryption of an entry, for reasonable block sizes, yields in practice hundreds of ciphertexts, which means that we have to perform hundreds of shuffle proofs. These are computable in polynomial-time but, even using the most efficient known solutions (e.g., [22,108]), not fast enough for practical purposes. This problem has been addressed in the literature but the known solutions typically reveal part of the permutation (e.g., [110]), which would break obliviousness and, thus, are not applicable in our setting.

General problem description. Let $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{E}, \mathsf{D}, \mathsf{Rnd})$ be a randomizable, additively homomorphic public-key encryption scheme with message space $\mathcal{M} = \mathbb{F}_p$ for some field $\mathbb{F}_p$, e.g., ElGamal [76] or Paillier [153]. Let furthermore $(\mathcal{P}, \mathcal{V})$ be a zero-knowledge proof system (ZKP) that takes as input two n-length ciphertext vectors $\vec{a} \in \mathsf{E}(ek, \vec{m})$ and $\vec{b} \in \mathsf{E}(ek, \pi(\vec{m}))$ where π is a permutation applied to some message vector $\vec{m}$, outputs a proof for the statement

$$\exists \vec{r}, \pi.\ \forall 1 \leq i \leq n.\ b_i = \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i).$$

The goal is to construct a proof system $(\mathcal{P}^*, \mathcal{V}^*)$ that takes as input two $n \times m$-dimensional ciphertext matrices $A \in \mathsf{E}(ek, M)$ and $B \in \mathsf{E}(ek, N)$ where N is M, column-wise permuted with a permutation π, and outputs a proof of the statement

$$\exists R, \pi.\ \forall 1 \leq i \leq n.\ \forall 1 \leq j \leq m.\ B_{i,j} = \mathsf{Rnd}(ek, A_{\pi^{-1}(i),j}, R_{i,j}).$$

We propose two solutions to this problem that we discuss in the sequel. The underlying idea of both solutions is to compress the data on which the shuffle proof is computed so as to lower the amount of proofs that have to be computed.

Algorithm 12 Batched zero-knowledge proofs of shuffle correctness.

```
Input of $\mathcal{P}^*$: $A$, $B$, $ek$, $\pi$, $R$
Input of $\mathcal{V}^*$: $A$, $B$, $ek$
1: $\mathcal{V}^*$ randomly selects $\vec{c} \leftarrow \{0,1\}^m$ and sends it to $\mathcal{P}^*$.
2: $\mathcal{P}^*$ computes for all $1 \leq i \leq n$ the partial ciphertext products
       $a_i = \bigotimes_{j=1}^{m} c_j A_{i,j}$ and $b_i = \bigotimes_{j=1}^{m} c_j B_{i,j}$
     and the corresponding partial randomness sum
       $r_i = \sum_{j=1}^{m} c_j R_{i,j}$
     where $c_j$ is the $j$-th bit of $\vec{c}$. $\mathcal{V}^*$ also computes $\vec{a}$ and $\vec{b}$.
3: $\mathcal{V}^*$ and $\mathcal{P}^*$ run $\mathcal{V}$ and $\mathcal{P}$, respectively, on $\vec{a}, \vec{b}, ek$ and $\vec{a}, \vec{b}, ek, \pi, \vec{r}$, respectively.
```

2.6.5.1. Batched Zero-Knowledge Proofs of Shuffle Correctness

The first solution is a new proof technique that we call *batched zero-knowledge proofs of shuffle correctness*, based on the idea of "batching" several instances and verifying them together. Our interactive protocol takes advantage of the homomorphic property of the public-key encryption scheme in order to batch the instances.

Intuitively, the batching algorithm randomly selects a subset of columns (i.e., block indices) and computes the row-wise homomorphic sum of the corresponding blocks for each row. It then computes the proof of shuffle correctness on the resulting single-block ciphertexts. The property we would like to achieve is that modifying even a single block in a row should lead to a different sum and, thus, be detected. Notice that naïvely multiplying all blocks together does not achieve the intended property, as illustrated by the following counterexample:

$$\begin{pmatrix} \mathsf{E}(pk,3) & \mathsf{E}(pk,4) \\ \mathsf{E}(pk,5) & \mathsf{E}(pk,2) \end{pmatrix} \qquad\qquad \begin{pmatrix} \mathsf{E}(pk,2) & \mathsf{E}(pk,5) \\ \mathsf{E}(pk,5) & \mathsf{E}(pk,2) \end{pmatrix}$$

In the above matrices, the rows have not been permuted but rather changed. Still, the row-wise sum is preserved, i.e., 7 in the first and 7 in the second. Hence, we cannot compute the sum over all columns. Instead, as proved in Appendix B.3, the intended property can be achieved with probability at least $\frac{1}{2}$ if each column is included in the homomorphic sum with probability $\frac{1}{2}$. Although a probability of $\frac{1}{2}$ is not sufficient in practice, repeating the protocol k times increases the probability to $(1 - \frac{1}{2^k})$.

The detailed construction is depicted in Algorithm 12. In line 12.1, $\mathcal{V}^*$ picks a challenge, which indicates which column to include in the homomorphic product. Upon receiving the challenge, in line 12.2, $\mathcal{P}^*$ and $\mathcal{V}^*$ compute the row-wise homomorphic sum of the columns indicated by the challenge. Finally, $\mathcal{V}^*$ and $\mathcal{P}^*$ run an off-the-shelf shuffle proof protocol between $\mathcal{V}$ and $\mathcal{P}$ on the resulting ciphertext lists (line 12.3). Finally, the protocol can be made non-interactive by using the Fiat-Shamir heuristic [78].

Formal guarantees. We establish the following result for our new protocol and prove it in Appendix B.3.

Algorithm 13 Shuffle proofs based on the hash-and-proof paradigm.

```
Input of $\mathcal{P}^*$: $A$, $B$, $ek$, $\pi$, $R$
Input of $\mathcal{V}^*$: $A$, $B$, $ek$
1: $\mathcal{V}^*$ randomly selects $\vec{z} \in \mathbb{F}_p^{m+2}$ and sends it to $\mathcal{P}^*$.
2: $\mathcal{P}^*$ computes for all $1 \leq i \leq n$ the partial ciphertext products
       $a_i = \mathsf{E}(ek, z_0; z_1) \bigotimes_{j=1}^{m} z_{j+1} A_{i,j}$ and $b_i = \mathsf{E}(ek, z_0; z_1) \bigotimes_{j=1}^{m} z_{j+1} B_{i,j}$
   and the corresponding randomness sum
       $r_i = \sum_{j=1}^{m} R_{i,j}$.
   $\mathcal{V}^*$ also computes $\vec{a}$ and $\vec{b}$.
3: $\mathcal{V}^*$ and $\mathcal{P}^*$ run $\mathcal{V}$ and $\mathcal{P}$, respectively, on $\vec{a}, \vec{b}$, $ek$ and $\vec{a}, \vec{b}, ek, \pi, \vec{r}$, respectively.
```

Theorem 2.2 (Batched zero-knowledge proofs of shuffle correctness). *Let Π_{PKE} be an additively homomorphic CPA-secure public-key encryption scheme and let $(\mathcal{P}, \mathcal{V})$ be a* ZKP *of shuffle correctness over Π_{PKE}. Then $(\mathcal{P}^*, \mathcal{V}^*)$ as defined in Algorithm 12 is a* ZKP *of shuffle correctness over Π_{PKE}, which is sound with probability at least $1/2$.*

2.6.5.2. The Hash-and-Proof Paradigm

The second solution improves the integrity proofs even further. The high-level idea is to design a compression technique that is collision-resistant with overwhelming probability, which is in contrast to the previous compression technique, which can lead to collisions with probability 1/2. In a nutshell, a technique based on pairwise independent hash functions [44] applied on each row of the ciphertext matrix allows for reducing the number of computed proofs of shuffle correctness to one.

The detailed construction is reported in Algorithm 13, where we leverage the fact that the message space $\mathcal{M} = \mathbb{F}_p$ for some field $\mathbb{F}_p$. Moreover, we let $\mathsf{E}(ek, z_0; z_1)$ denote the encryption of z_0 with key ek and randomness z_1. In line 13.1, $\mathcal{V}^*$ picks a challenge, which can be seen as the coefficients of the pairwise independent hash function. Upon receiving the challenge, in line 13.2, $\mathcal{P}^*$ and $\mathcal{V}^*$ compute the row-wise homomorphic sum of the columns as dictated by the challenge, additionally adding the encryption of z_0 using the randomness z_1 for both $\vec{a}$ and $\vec{b}$. Finally, $\mathcal{V}^*$ and $\mathcal{P}^*$ run an off-the-shelf shuffle proof protocol between $\mathcal{V}$ and $\mathcal{P}$ on the resulting ciphertext lists (line 13.3). As before, the protocol can be made non-interactive by applying the Fiat-Shamir heuristic [78].

Formal guarantees. We establish the following result for our new protocol and prove it in Appendix B.3.

Theorem 2.3 (Hash-and-Proof). *Let Π_{PKE} be an additively homomorphic CPA-secure public-key encryption scheme and let $(\mathcal{P}, \mathcal{V})$ be a* ZKP *of shuffle correctness over Π_{PKE}. Then $(\mathcal{P}^*, \mathcal{V}^*)$ is a* ZKP *of shuffle correctness over Π_{PKE}.*

2.6.5.3. Discussion

The improvements presented in this section cannot only be applied to GORAM, they can also be used to speed-up PIR-GORAM. The reason is that in the flush algorithm, clients have to prove that they either randomized an entry or that they know a signing key allowing them to change the data. The first half of this proof is clearly subject to the same problems as the ones we face in GORAM: entries with bigger block sizes require more than one public-key ciphertext and, consequently, the number of proofs to be computed is linear in the block size. Inspecting the new proof techniques, we observe that if π is the identity function and the number of rows in the matrices is one, then we have reproduced exactly the setting of PIR-GORAM for a single such proof. Hence, batched proofs of shuffle correctness and the hash-and-proof paradigm can also be applied to general plaintext-equivalence-proofs (PEPs).

2.7. GORAM with Accountable Integrity (A-GORAM)

In this section we relax the integrity property by introducing the concept of accountability. In particular, instead of letting the server check the correctness of client operations, we develop a technique that allows clients to detect *a posteriori* non-authorized changes on the database and blame the misbehaving party. Intuitively, each entry is accompanied by a tag (technically, a chameleon hash along with the randomness corresponding to that entry), which can only be produced by clients having write access. All clients can verify the validity of such tags and, eventually, determine which client inserted an entry with an invalid tag. This makes the construction more efficient and scalable, significantly reducing the computational complexity both on the client and on the server side, since zero-knowledge proofs are no longer necessary and, consequently, the outermost encryption can be implemented using symmetric, as opposed to asymmetric, cryptography. Such a mechanism is supposed to be paired with a data versioning protocol in order to avoid data losses: as soon as one of the clients detects an invalid entry, the misbehaving party is punished and the database is reverted to the last safe state (i.e., a state where all entries are associated with a valid tag).

2.7.1. Cryptographic preliminaries

Our construction relies on predicate encryption, private-key encryption, digital signatures, and chameleon hashes. We summarize the missing notation for private-key encryption and chameleon hashes in Table 2.4. A chameleon hash allows for computing an explicit collision for the hash value if one knows a secret trapdoor. Further details are postponed to Appendix A.

Primitive	**Notation**
Private-key encryption	Π_{SE}
Key generation	$k \leftarrow \mathsf{Gen}_{\mathsf{SE}}(1^\lambda)$
Encryption	$c \leftarrow \mathcal{E}(k, m)$
Decryption	$m \leftarrow \mathcal{D}(k, c)$
Chameleon hash	Π_{CH}
Key generation	$(cpk, csk) \leftarrow \mathsf{Gen}_{\mathsf{CHF}}(1^\lambda)$
Hashing	$t \leftarrow \mathsf{CH}(cpk, m, r)$
Collision	$r' \leftarrow \mathsf{Col}(csk, m, r, m')$ such that $\mathsf{CH}(cpk, m, r) = \mathsf{CH}(cpk, m', r')$

Table 2.4.: Notation for cryptographic primitives.

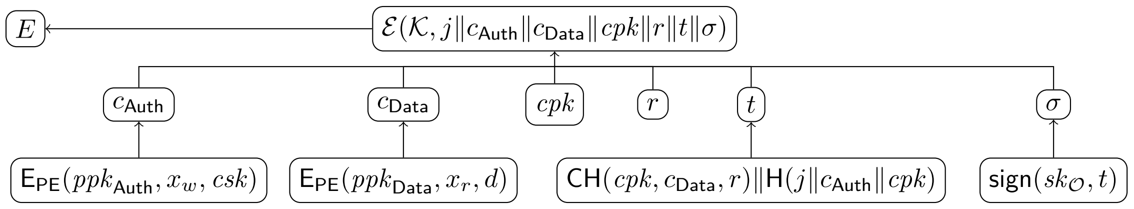

E
$\mathcal{E}(\mathcal{K}, j\|c_{\mathsf{Auth}}\|c_{\mathsf{Data}}\|cpk\|r\|t\|\sigma)$
c_{Auth}
c_{Data}
cpk
r
t
σ
$\mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_w, csk)$
$\mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Data}}, x_r, d)$
$\mathsf{CH}(cpk, c_{\mathsf{Data}}, r)\|\mathsf{H}(j\|c_{\mathsf{Auth}}\|cpk)$
$\mathsf{sign}(sk_{\mathcal{O}}, t)$

Figure 2.8.: The entry structure of an entry in the database.

2.7.2. System Assumptions

Data structures and database layout. We assume the same layout and data structures as for GORAM. Additionally, we use a log file Log so as to detect who has to be held accountable in case of misbehavior. Log is append-only and consists of the list of paths uploaded to the server, each of them signed by the respective client. Such an append-only log file can be realized both in a centralized [99] or decentralized way [62].

Client capabilities. As in GORAM, every client $\mathcal{C}_i$ holds a capability cap_i containing a collection of keys: predicate encryption keys $psk^{\mathsf{Auth}}_{f_i}$ and $psk^{\mathsf{Data}}_{f_i}$ and a key $\mathcal{K}$ for the top level encryption, which is now, however, replaced by a private-key version.

Structure of an entry and access control modes. The structure of an entry in the database is depicted in Figure 2.8. An entry E is protected by a top-level private-key encryption scheme with a key $\mathcal{K}$ that is shared by the data owner $\mathcal{O}$ and all clients $\mathcal{C}_1, \dots, \mathcal{C}_n$. Under the encryption, E contains several elements, which we explain below:

- j is the index of the entry;
- c_{Auth} is a predicate encryption ciphertext that encrypts the private key csk of a chameleon hash function under an attribute x_w, which regulates the write access;
- c_{Data} is unchanged;
- cpk is the public key of a chameleon hash function, i.e., the counterpart of csk encrypted in c_{Auth};

- r is some randomness used in the computation of t;
- t is a concatenation of hash tags: a chameleon hash tag produced by hashing c_{Data} under randomness r, and a normal hash tag produced by hashing j, c_{Auth}, and the public key of the chameleon hash function cpk; only c_{Data} is hashed with the chameleon hash function so as to not allow clients to change every other part but c_{Data};
- σ is a signature on the tag t, signed by the data owner $\mathcal{O}$.

Intuitively, only clients with write access are able to decrypt c_{Auth}, and thus to retrieve the key csk required to compute a collision for the new entry d' (i.e., to find a randomness r' such that the chameleon hash t for the old entry d and randomness r is the same as the one for d' and r'). The fundamental observation is that the modification of an entry is performed without changing the respective tag. Consequently, the signature σ is the same for the old and for the new entry. Computing a collision is the only way to make the tag t, originally signed by the data owner, a valid tag also for the new entry d'. Therefore verifying the signature and the chameleon hash suffices to make sure that the entry has been only modified by authorized clients.

2.7.3. Construction

Basic Algorithms. The basic algorithms follow the ones defined in Section 2.6.3, except for natural adaptions to the new entry structure. Furthermore, the zero-knowledge proofs are no longer computed and the rerandomization steps are substituted by re-encryptions. Finally, clients upload on the server signed paths, which are stored in the Log. We detail the differences to the algorithms of GORAM in Section B.5.

Entry Verification. We introduce an auxiliary verification function that clients run in order to verify the integrity of an entry. During the execution of any protocol we maintain the invariant that, whenever a client i (or the data owner himself) parses an entry j that she downloaded from the server, she executes Algorithm 14. If the result is $\perp$, then the client runs $\mathsf{blame}(cap_i, \mathsf{Log}, j)$. The client also runs $\mathsf{blame}(cap_i, \mathsf{Log}, j)$ if she does not find j on the downloaded path even if the index mapping says so.

Algorithm 14 The pseudo-code for the verification of an entry in the database which is already decrypted.

```
Input: An entry $(j, c_{\mathsf{Auth}}, c_{\mathsf{Data}}, r, cpk, t, \sigma)$ and the verification key $vk_{\mathcal{O}}$ of $\mathcal{O}$.
Output: $\top$ if verification succeeds, $\perp$ otherwise.
1: if $t = \mathsf{CH}(cpk, c_{\mathsf{Data}}, r) \| \mathsf{H}(j \| c_{\mathsf{Auth}} \| cpk)$ and $\top = \mathsf{vfy}(\sigma, vk_{\mathcal{O}}, t)$ then return $\top$
2: else return $\perp$
3: end if
```

Blame. In order to execute the function $\mathsf{blame}(cap_i, \mathsf{Log}, j)$, the client must first retrieve Log from the server. Afterwards, she parses backwards the history of modifications by decrypting the paths present in the Log. The client stops only when she finds the desired entry indexed by j in a consistent state, i.e., the data hashes to the associated tag t and the signature is valid. At this point the client moves forwards on the Log until she finds an uploaded path where the entry j is supposed to lay on (the entry might be associated with an invalid tag or missing). The signature on the path uniquely identifies the client, whose identity is added to a list L of misbehaving clients. Finally, all of the other clients that acknowledged the changes of the inconsistent entry are also added to L, since they did not correctly verify its chameleon signature. If the entry is missing, we only add the client who removed it to L. No other client can be deemed malicious since a missing entry is only detected when a client tries to actively read or write it.

2.7.4. Discussion

As explained above, the accountability mechanism allows for the identification of misbehaving clients with a minimal computational overhead in the regular clients' operation. However, it requires the server to store a log that is linear in the number of modifications to the database and logarithmic in the number of entries. This is required to revert the database to a safe state in case of misbehaviour. Consequently, the blame algorithm results expensive in terms of computation and communication with the server, in particular for the entries that are not regularly accessed. Nonetheless, blame is supposed to be only occasionally executed, therefore we believe this design is acceptable in terms of service usability. Furthermore, we can require all the parties accessing the database to synchronize on a regular basis so as to verify the content of the whole database and to reset the Log, in order to reduce the storage on the server side and, thus, the amount of data to transfer in the blame algorithm. Such an approach could be complemented by an efficient versioning algorithm on encrypted data, which is however beyond the scope of this work and left as a future work. Finally, we also point out that the accountable-integrity property targeted by A-GORAM sacrifices anonymity, since users have to sign the paths they upload to the server. This issue can be easily overcome by using any anonymous credential system that supports revocation [38].

2.8. Scalable Solution (S-GORAM)

Even though the personal record management systems we consider rely on simple client-based read and write permissions, the predicate encryption scheme used in GORAM and A-GORAM support in principle a much richer class of access control policies, such as role-based access control (RBAC) or attribute-based access control (ABAC) [113]. If we stick to client-based read and write permissions, however, we can achieve a more efficient construction that scales to thousands of clients. To this end, we replace the predicate encryption scheme with a broadcast encryption scheme [85], which guarantees that a specific subset of clients is able to decrypt a given ciphertext. This choice affects the entry structure as follows (cf. Figure 2.8):

Property	PIR-GORAM	TAO-GORAM	GORAM	A-GORAM	S-GORAM
Secrecy	✓	✓	✓	✓	✓
Integrity	✓	✓	✓	Accountable	Accountable
Tamper-resistance	✓	✓	✓	✗	✗
Obliviousness ($\mathcal{C} + \mathcal{S}$)	✓	✓	✗	✗	✗
Obliviousness ($\mathcal{S}$ only)	✓	✓	✓	✓	✓
Anonymity	✗	✗	✓	✗	✗
Access control	R/W	R/W	ABAC	ABAC	R/W

Table 2.5.: Security and privacy properties achieved by each construction.

- c_{Data} is the broadcast encryption of d;
- c_{Auth} is the broadcast encryption of csk.

The subset of clients that can decrypt c_{Data} (resp. c_{Auth}) is then set to be the same subset that holds R (resp. RW) permissions on the given entry. By applying the aforementioned modifications on top of A-GORAM, we obtain a much more efficient and scalable instantiation, called S-GORAM, that achieves a smaller constant in the computational complexity (linear in the number of clients). For more details on the performance evaluation and a comparison with A-GORAM, we refer to Section 2.10.

2.9. Security and Privacy Results

In this section, we show that the Group ORAM instantiations presented in Section 2.4, in Section 2.5, in Section 2.6, in Section 2.7, and in Section 2.8 achieve the security and privacy properties stated in Section 2.2.3. The proofs are reported in Appendix B.4. A brief overview of the properties guaranteed by each construction is shown in Table 2.5. As previously discussed, relaxing the obliviousness property so as to consider only security against the server or assuming a trusted component in the system is required to enable constructions that are more efficient from a communication point of view. Hence, TAO-GORAM and GORAM are optimal with respect to communication. As opposed to TAO-GORAM, which is oblivious with respect to malicious clients, GORAM does not assume any trusted component in the system. Furthermore, dropping the computationally expensive integrity checks in favor of an accountability mechanism is crucial to achieve computational efficiency. It follows that A-GORAM and S-GORAM provide accountable integrity as opposed to integrity and tamper resistance. Having an accountable system trivially implies the loss of anonymity, as defined in Definition 2.7, although it is still possible to achieve pseudonym-based anonymity by employing anonymous credentials. The other privacy properties of our system, namely secrecy and obliviousness, are fulfilled by all of our instantiations. Moreover, by replacing predicate encryption with broadcast encryption (S-GORAM), we sacrifice the possibility to enforce ABAC policies, although we can still handle client-based read/write permissions.

The following theorems characterize the security and privacy properties achieved by each cryptographic instantiation presented in this chapter. Interestingly enough, the properties of TAO-GORAM are only conditioned by the trusted-component-based parallel ORAM construction that we base it on since our additions are not of cryptographic nature but simply modify the software component that synchronizes client accesses.

Theorem 2.4 (PIR-GORAM)**.** *2.4.1. Let Π_{PKE} be a CPA-secure encryption scheme, then* PIR-GORAM *achieves secrecy.*

2.4.2. Let Π_{DS} be an existentially unforgeable digital signature scheme, ZKP *be a zero-knowledge proof of knowledge protocol, and Π_{PKE} be a CCA-secure encryption scheme, then* PIR-GORAM *achieves integrity.*

2.4.3. Let Π_{DS} be an existentially unforgeable digital signature scheme and let Π_{PKE} be a CCA-secure encryption scheme, then PIR-GORAM *achieves tamper resistance.*

2.4.4. Let Π_{PIR} be a private information retrieval scheme, let Π_{PKE} be a CPA-secure encryption scheme, let Π_{DS} be an existentially unforgeable digital signature scheme, and let ZKP *be a zero-knowledge proof of knowledge protocol, then* PIR-GORAM *is oblivious against malicious clients.*

Theorem 2.5 (TAO-GORAM)**.** *Assume that TaoStore is a secure realization of a parallel ORAM. Then* TAO-GORAM *achieves secrecy, integrity, tamper resistance, and obliviousness against malicious clients.*

Theorem 2.6 (GORAM)**.** *2.6.1. Let Π_{PE} be an attribute-hiding predicate encryption scheme. Then* GORAM *achieves secrecy.*

2.6.2. Let ZKP *be a zero-knowledge proof system. Then* GORAM *achieves integrity.*

2.6.3. Let ZKP *be a zero-knowledge proof system and Π_{PE} be an attribute-hiding predicate encryption scheme. Then* GORAM *achieves tamper-resistance.*

2.6.4. Let ZKP *be a zero-knowledge proof system and Π_{PKE} be a CPA-secure public-key encryption scheme. Then* GORAM *achieves obliviousness.*

2.6.5. Let ZKP *be a zero-knowledge proof system. Then* GORAM *achieves anonymity.*

Theorem 2.7 (A-GORAM)**.** *2.7.1. Let Π_{PE} be an attribute-hiding predicate encryption scheme. Then* A-GORAM *achieves secrecy.*

2.7.2. Let Π_{CH} be a collision-resistant, key-exposure free chameleon hash function, Π_{DS} be an existentially unforgeable digital signature scheme, and Π_{PE} be an attribute-hiding predicate encryption scheme. Then A-GORAM *achieves accountable integrity.*

2.7.3. Let Π_{SE} be a CPA-secure private-key encryption scheme. Then A-GORAM *achieves obliviousness.*

Theorem 2.8 (S-GORAM)**.** *2.8.1. Let Π_{BE} be an adaptively secure broadcast encryption scheme and Π_{SE} be a CPA-secure private-key encryption scheme. Then* S-GORAM *achieves secrecy.*

Scheme	Instantiation
Π_{SE}	AES [63]
Π_{PKE}	ElGamal [76], Cramer-Shoup [60]
Π_{PE}, Π_{POE}	Katz *et al.* [113]
Π_{BE}	Gentry and Waters [85]
Π_{PIR}	XPIR [2]
ZKP	OR-proof [58], plaintext-equivalence-proof [108], DLog [173], Groth-Sahai [94], proofs of shuffle correctness [22]
Π_{CH}	Nyberg and Rueppel [13]
Π_{DS}	RSA [165]

Table 2.6.: Cryptographic scheme instantiations.

2.8.2. Let Π_{CH} *be a collision-resistant, key-exposure free chameleon hash function ,* Π_{DS} *be an existentially unforgeable digital signature scheme,* Π_{BE} *be an adaptively secure broadcast encryption scheme, and* Π_{SE} *be a CPA-secure private-key encryption scheme. Then* A-GORAM *achieves accountable integrity.*

2.8.3. Let Π_{SE} *be a CPA-secure private-key encryption scheme. Then* A-GORAM *achieves obliviousness.*

2.10. Implementation and Experiments

In this section, we present the concrete instantiations of the cryptographic primitives that we previously described (Section 2.10.1) and we describe our implementation and discuss the experimental evaluation (Section 2.10.2).

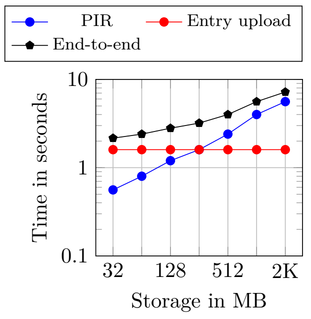
2.10.1. Cryptographic Instantiations

We summarize the concrete instantiations in Table 2.6 and postpone detailed considerations to Appendix B.1.

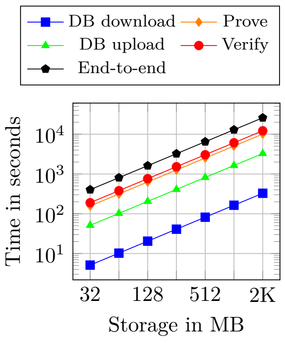
2.10.2. Experiments

We implemented the six different versions of GORAM in Java (PIR-GORAM, GORAM with off-the-shelf shuffle proofs, batched shuffle proofs, and shuffle proofs based on the hash-and-proof paradigm, A-GORAM, and S-GORAM). Furthermore, we also implemented A-GORAM and S-GORAM on Amazon EC2. For the zero-knowledge proofs computed on predicate encryption, we build on a library [15] that implements Groth-Sahai proofs [94], which internally relies on jPBC/PBC [42, 137].

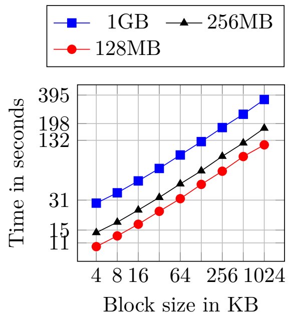
Cryptographic setup. We use MNT curves [147] based on prime-order groups for primes of length 224 bits. This results in 112 bits of security according to different organizations [32].



(a) Access without flush, 1MB block size.

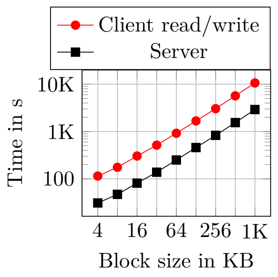


(b) Flush, 1MB block size.

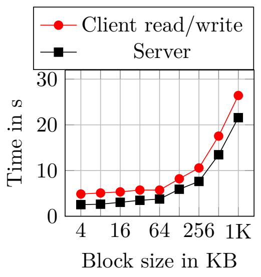


(c) Amortized time, varying block size.

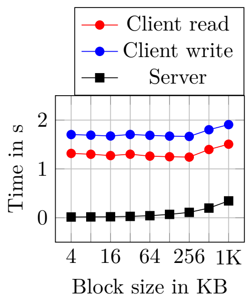
Figure 2.9.: The end-to-end running time of an operation in PIR-GORAM.



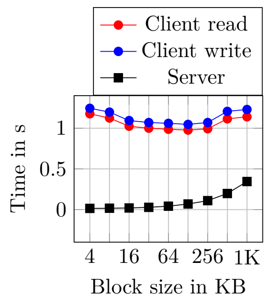
(a) GORAM.



(b) GORAM with batched shuffle proofs and $k=3$.



(c) A-GORAM.

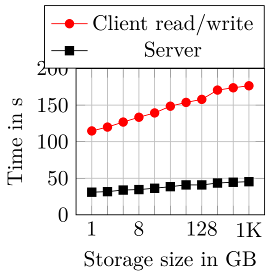


(d) S-GORAM.

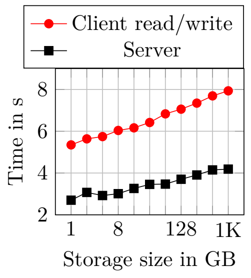
Figure 2.10.: The average execution time for the read and write protocol on client and server for varying B where $BN = 1\text{GB}$ and $G = 4$.

We deploy AES with 128 bit keys and we instantiate the ElGamal and Cramer-Shoup encryption scheme, the RSA signature scheme, and the chameleon hash function with a security parameter of 2048 bits. According to NIST [32], this setup is secure until 2030.

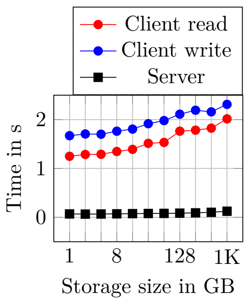
Micro-benchmarks. We evaluated the six different implementations. As a first experiment, we measured the computation times on client and server for the read and write operation for the constructions without accountable integrity. We performed these experiments on an Intel Xeon with 8 cores and 2.60GHz in order to show the efficiency gained by using batched shuffle proofs instead of off-the-shelf zero-knowledge proofs of shuffle correctness. We vary different parameters: the database size from 128MB to 2GB (PIR-GORAM) and from 1GB to 1TB (GORAM, A-GORAM, and S-GORAM), the block size from 4KB to 1MB, the number of clients from 1 to 10, the number of cores from 1 to 8, and for batched shuffle proofs also the number of iterations k from 1 to 128. For GORAM, A-GORAM, and S-GORAM we fix a bucket size of 4 since Stefanov *et al.* [187] showed that this value is



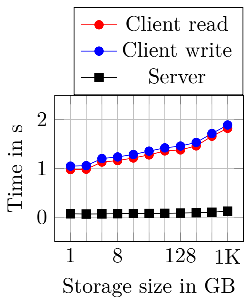
(a) GORAM with B = 4KB.



(b) GORAM with batched shuffle proofs, $B = 4$KB, and $k = 4$.



(c) A-GORAM with $B =$ 128KB.



(d) S-GORAM with $B =$ 128KB.

Figure 2.11.: The average execution time for the read and write protocol on client and server for varying BN where $G = 4$.

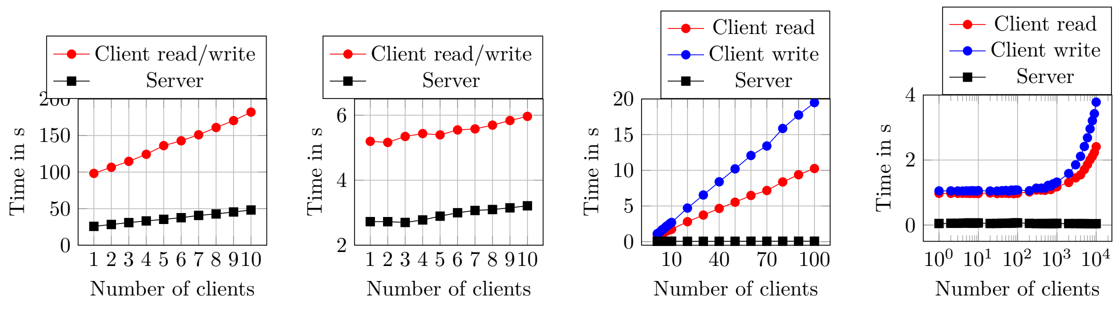
sufficient to prevent buckets from overflowing.

The second experiment focuses on the solution with accountability. Here we measure also the overhead introduced by our realization with respect to a state-of-the-art ORAM construction, i.e., the price we pay to achieve a wide range of security and privacy properties in a multi-client setting. Another difference from the first experiment is the hardware setup. We run the server side of the protocol in Amazon EC2 and the client side on a MacBook Pro with an Intel i7 and 2.90GHz. We vary the parameters as in the previous experiment, except for the number of clients which we vary from 1 to 100 for A-GORAM and from 1 to 10000 for S-GORAM, and the number of cores which are limited to 4. In the experiments where the number of cores is not explicitly varied, we use the maximum number of cores available.

2.10.3. Discussion

Figure 2.9 and Figure 2.14 report the results for PIR-GORAM. Figure 2.9a shows the end-to-end and partial running times of an access to the ORAM when the flush algorithm is not executed, whereas Figure 2.9b depicts the worst case running time (i.e., with flush operation). We assume a mobile LTE connection for the network, i.e., 100Mbit/s downlink and 50Mbit/s uplink in peak. For the example of the medical record which usually fits into 128MB (resp. 256MB for additional files such as X-ray images), the amortized times per access range from 11 (resp. 15) seconds for 4KB up to 131 (resp. 198) seconds for 1MB sized entries (see Figure 2.9c).

Figure 2.14 shows the improvement as we compare the combined proof computation and proof verification time in the flush algorithm of PIR-GORAM, first as described in Section 2.4 and then with the integrity proof based on the hash-and-proof paradigm (see Section 2.6.5.2). We observe that our expectations are fulfilled: the larger the block size, the more effect has the hash computation since the number of proofs to compute decreases.



(a) GORAM with B = 4KB.

(b) GORAM with batched shuffle proofs, B = 4KB, and $k = 4$.

(c) A-GORAM with B = 128KB.

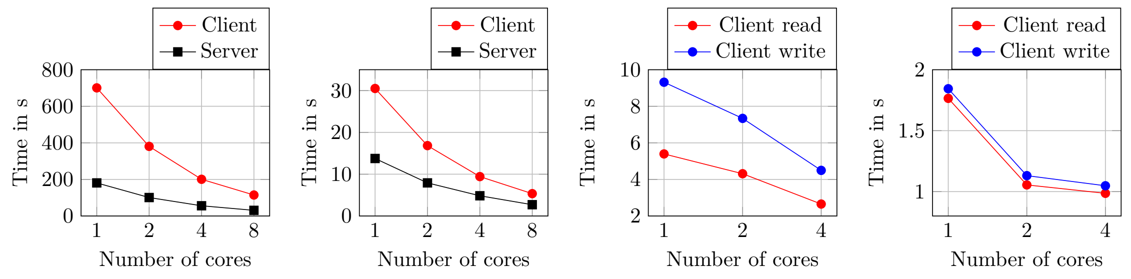
(d) S-GORAM with B = 128KB.

Figure 2.12.: The average execution time for the read and write protocol on client and server for varying G where BN = 1GB.

Concretely, with 1MB block size we gain a speed-up of about 4% for flush operations with respect to the construction without homomorphic hash. 4% does not sound much and the reason for this little improvement is quite straightforward: the computations performed for computing the hash function and those performed for computing the single PEPs are the same. The only difference lies in the verification of the proof, which incurs one more modular exponentiation than the recomputation of the hash on the server side. Hence, there is an improvement, but this improvement shows only for big block sizes, say, more than 1MB. We will see further below, it has much more effect on GORAM.

Our solution TAO-GORAM only adds access control to the actual computation of TaoStore's trusted proxy [170]. Interestingly enough, TaoStore's bottleneck is not computation, but communication. Hence, our modifications do not cause any noticeable slowdown on the throughput of TaoStore. Consequently, we end up with a throughput of about 40 operations per second when considering an actual deployment of TAO-GORAM in a cloud-based setting [170].

The results of the experiments for GORAM, A-GORAM, and S-GORAM are reported in Figure 2.10–2.17. As shown in Figure 2.10a, varying the block size has a linear effect in the construction without batched shuffle proofs. As expected, the batched shuffle proofs improve the computation time significantly (Figure 2.10b). The new scheme even seems to be independent of the block size, at least for block sizes less than 64KB. This effect is caused by the parallelization. Still, the homomorphic multiplication of the public-key ciphertexts before the batched shuffle proof computation depends on the block size (line 12.2). We do not depict individual results for GORAM with proofs based on the hash-and-proof paradigm: the computation necessary is almost equivalent to that for batched shuffle proofs with $k = 1$; the only difference being the homomorphic pre-computation which is slightly more expensive. Hence, whenever we state something about batched shuffle proofs, the same holds for the hash-and-proof paradigm. Figure 2.10c and Figure 2.10d show the results for A-GORAM and S-GORAM. Since the computation time is in practice almost



(a) GORAM with $B = 4$KB, and $G = 4$.
(b) GORAM with batched shuffle proofs, $B = 4$KB, $G = 4$, and $k = 4$.
(c) A-GORAM with $BN = 1$GB, $B = 128$KB, and $G = 20$.
(d) S-GORAM with $BN = 1$GB, $B = 128$KB, and $G = 20$.

Figure 2.13.: The average execution time for the read and write protocol on client and server for a varying number of cores where $BN = 1$GB.

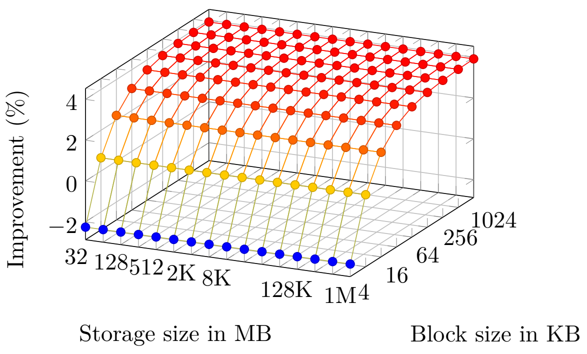


Figure 2.14.: The improvement in percent when comparing the combined proof computation time on the client and proof verification time on the server for varying storage and block sizes, once without and once with the universal homomorphic hash.

independent of the block size, we can choose larger block sizes in the case of databases with large files, thereby allowing the client to read (resp. write) a file in one shot, as opposed to running multiple read (resp. write) operations. We identify a minimum computation time for 128KB as this is the optimal trade-off between the index map size and the path size. The server computation time is low and varies between 15ms and 345ms, while client operations take less than 2 seconds for A-GORAM and less than 1.3 seconds for S-GORAM. As we obtained the best results for 4KB in the experiments for GORAM and 128KB for the others, we use these block sizes in the sequel.

The results obtained by varying the storage size (Figure 2.11) and the number of clients (Figure 2.12) prove what the computational complexity suggests. Nevertheless, it is interesting to see the tremendous improvement in computation time between GORAM with and without batched shuffle proofs. The results obtained by varying the iteration time of the batched shuffle proof protocol are depicted in Figure 2.15 and we verify the expected linear dependency. Smaller values of k are more efficient but higher values give a better

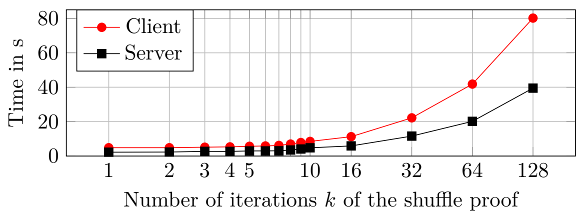


Figure 2.15.: Average execution time for the read and write protocol on client and server for GORAM with batched shuffle proofs and varying k where $BN = 1$GB, $B = 8$KB, and $G = 4$.

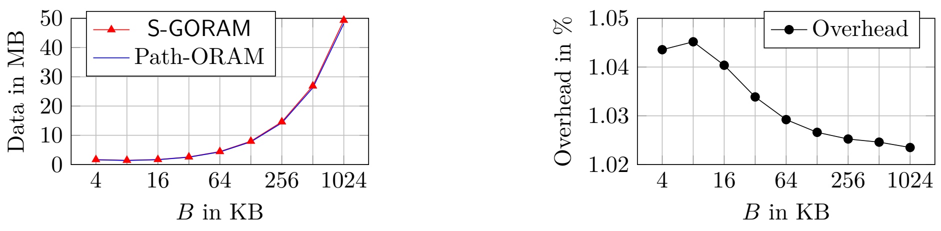


Figure 2.16.: The up-/download amount of data compared between Path-ORAM [187] and S-GORAM for varying B while $BN = 1$GB and $G = 4$.

soundness probability. Even more impressive, the technique based on the hash-and-proof paradigm speeds up GORAM even further. As shown in Figure 2.17a, we gain one order of magnitude (14x on the client and 10.8x on the server for $k = 128$) since it is sufficient to compute a single proof rather than k. Notice that we do not gain an improvement of 128 since the shuffle proofs are just one component of the read and write operations.

If we compare A-GORAM and S-GORAM in Figure 2.12c and Figure 2.12d we can see that S-GORAM scales well to a large amount of users as opposed to A-GORAM. The good scaling behavior is due to the used broadcast encryption scheme: it only computes a constant number of pairings independent of the number of users for decryption while the opposite holds for predicate encryption. Nevertheless, we identify a linear growth in the times for S-GORAM, which arises from the linear number of exponentiations that are computed. For instance, in order to write 128KB in a 1GB storage that is used by 100 users, A-GORAM needs about 20 seconds while S-GORAM only needs about 1 second. Even when increasing the number of users to 10000, S-GORAM requires only about 4 seconds, a time that A-GORAM needs for slightly more than 10 users.

Figure 2.13 shows the results obtained by varying the number of cores. In GORAM most of the computation, especially the zero-knowledge proof computation, can be easily parallelized. We observe this fact in both results (Figure 2.13a and Figure 2.13b). In the efficient construction we can parallelize the top-level encryption and decryption, the verification of the entries, and the predicate ciphertext decryption. Also in this case parallelization significantly improves the performance (Figure 2.13c and Figure 2.13d). Notice that we run the experiments in this case for 20 clients, as opposed to 4 as done for the other constructions, because the predicate ciphertext decryption takes the majority of the computation time and, hence, longer ciphertexts take longer to decrypt and the

Construction	Client time	Server time
GORAM with $k = 128$	91.315 s	39.213 s
GORAM with HaP	5.980 s	3.384 s
Improvement	**14x**	**10.8x**

(a) Comparison of GORAM with batched shuffle proofs and GORAM with shuffle proofs based on hash-and-proof (HaP) for 10 users, 1GB storage, and 8KB block size.

Scheme	Client Read	Client Write	Server
S-GORAM	0.981s	1.075s	0.068s
Path-ORAM	0.042s	0.042s	0.002s

(b) Comparison of the computation times between Path-ORAM [187] (single-client!) and S-GORAM on 1GB storage size, 128KB block size and 100 clients.

Figure 2.17.: Comparison of the two integrity proof improvements and the overhead with respect to state-of-the-art ORAM.

parallelization effect can be better visualized.

Finally, Figure 2.17b compares S-GORAM with the underlying Path-ORAM protocol. Naturally, since Path-ORAM only uses symmetric encryption, no broadcast encryption, and no verification with chameleon signatures, the computation time is much lower. However, the bottleneck of both constructions is actually the amount of data that has to be downloaded and uploaded by the client (Figure 2.16). The time required to upload and download data may take much more time than the computation time, given today's bandwidths. Here the overhead is only between 1.02% and 1.05%. For instance, assuming a mobile client using LTE (100Mbit/s downlink and 50Mbit/s uplink in peak) transferring 2 and 50 MB takes 480ms and 12s, respectively. Under these assumptions, considering a block size of 1MB, we get a combined computation and communication overhead of 8% for write and 7% for read, which we consider a relatively low price to pay to get a wide range of security and privacy properties in a multi-client setting.

2.11. Case Study: Personal Health Records

We briefly discuss a potential application of GORAM, namely, a privacy-preserving personal health record (PHR) management system. As the patient should have the control of her own record, the patient is the data owner. The server is some cloud storage provider, which may be chosen by the patient or directly by the state for all citizens (e.g., ELGA in Austria). The healthcare personnel (doctors, nurses, pharmacies, and so on) constitutes the clients.

We discuss now possible real-world attacks on PHRs and how the usage of GORAM prevents them. One typical threat is the cloud provider trying to learn customer information (e.g., to sell it or to use it for targeted advertising). For instance, as previously discussed, monitoring the accesses to DNA sequences would allow the service provider to learn the patient's disease: these kinds of attacks are not possible because of obliviousness and data secrecy. PIR-GORAM and TAO-GORAM can even protect against such an attack if the cloud storage provider colludes with one of the healthcare personnel. Another possible attack could be a pharmacy that tries to increase its profit by changing a prescription for a cheap medicine into one that prescribes an expensive medicine. However, pharmacies would not have write access to prescriptions, and hence, these cannot be changed or, in A-GORAM and S-GORAM, the misbehaving pharmacy can be blamed by the data owner. A common procedure in order to sign a contract with a health insurance is the health check. The patient might want to hide health information from the insurance in order to get a lower fee. To this end, the patient could simply try to drop this information. Dropping of entries in the database is, however, either prevented by making such documents permanent or, in A-GORAM and S-GORAM, by letting the insurance, who sees that some documents are missing, blame the patient. Using the backup strategy, the missing documents can be restored.

Finally, while PIR-GORAM and TAO-GORAM offer strong privacy guarantees at a certain price and under specific trust assumptions, respectively, we think that GORAM with batched shuffle proofs or shuffle proofs based on the hash-and-proof paradigm (even more so A-GORAM and S-GORAM) is a practical solution for the management of today's PHRs, since they are of rather small size. For instance, the data today stored in e-health cards is at most 128KB. The current trend is to store the remaining medical information (e.g., DNA information) on an external server, which can be accessed by using the card. This is exactly our setting, except that we allow for accessing PHRs even without the card, which is crucial in emergency situations. DNA information takes approximately 125MB[8] [166] and all our constructions offer an adequate performance for databases of a few gigabytes, with A-GORAM and S-GORAM performing better for the retrieval of large amounts of data, thanks to the possibility of using larger block sizes.

2.12. Related Work

Oblivious RAM. Oblivious RAM (ORAM) [88] is a technique originally devised to protect the access pattern of software on the local memory and thus to prevent the reverse engineering of that software. The observation is that encryption by itself prevents an attacker from learning the content of any memory cell but monitoring how memory is accessed and modified may still leak a great amount of sensitive information. While the first constructions were highly inefficient [88], recent groundbreaking research paved the way for a tremendous efficiency boost, exploiting ingenious tree based constructions [3, 8, 41, 65, 66, 91, 138, 159, 178, 183, 185], server side computations [103, 146], and trusted hardware [29, 106, 136, 170, 184].

[8]The actual DNA sequence takes about 200GB but one usually shares only the mutations, i.e., the differences of the considered genome to the average human genome. These mutations are only 0.1% of the overall sequence.

Work	MC	MD	PI	CS	Pr	AC	P	S comp.	C comp.	Comm.
Franz *et al.* [81]	✓	✗	✓	✗	✗	✓	✗	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
(A-/S-)GORAM (§ 2.6/2.7/2.8)	✗	✗	✗	✓	✗	✓	✗	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
PIR-GORAM (§ 2.4)	✓	✓	✗	✓	✗	✓	✗	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$
BCP [37]	✗	✗	✓	✓	✗	✗	✓	$\Omega(\log^3(n))$	$\Omega(\log^3(n))$	$\Omega(\log^3(n))$
CLT [50]	✗	✗	✓	✓	✗	✗	✓	$O(\log^2(n))$	$O(\log^2(n))$	$O(\log^2(n))$
PrivateFS [193]	✗	✗	✗	✓	✗	✗	✓	$O(\log^2(n))$	$O(1)$	$O(\log^2(n))$
Shroud [136]	✗	✗	✗	✗	✓	✗	✓	$O(\log^2(n))$	$O(1)$	$O(\log^2(n))$
TaoStore [170]	✗	✗	✗	✗	✓	✗	✓	$O(\log(n))$	$O(1)$	$O(\log(n))$
TAO-GORAM (§ 2.5)	✓	✓	✗	✗	✓	✓	✓	$O(\log(n))$	$O(1)$	$O(\log(n))$

Table 2.7.: Comparison of the related work supporting multiple clients to our constructions. The abbreviations mean: **MC**: oblivious against malicious clients, **MD**: supports multiple data owners sharing their data in one ORAM, **PI**: requires the periodic interaction with the data owner, **CS**: requires synchronization among clients, **AC**: access control, **Pr**: trusted proxy, **P**: parallel accesses, **S comp.**: server computation complexity, **C comp.**: client communication complexity, **Comm.**: communication complexity.

While a few ORAM constructions guarantee the integrity of user data [185, 192], none of them is suitable to share data with potentially distrustful clients. Goodrich *et al.* [92] studied the problem of multi-client ORAM, but their attacker model does not include malicious, and potentially colluding, clients. Furthermore, their construction does not provide fine-grained access control mechanisms, i.e., either all members of a group have access to a certain data, or none has. Finally, this scheme does not allow the clients to verify the data integrity.

The fundamental problem in existing ORAM constructions is that all clients must have access to the ORAM key, which allows them to read and potentially disrupt the entire database. Hence, dedicated solutions tailored to the multi-client setting are required.

Multi-client ORAM. A few recent constructions gave positive answers to the question of whether multi-client ORAM can be built, devising ORAM constructions in the multi-client setting, which specifically allow the data owner to share data with other clients while imposing fine-grained access control policies. Although, at a first glance, these constructions share the same high-level goal, they actually differ in a number of important aspects. Therefore we find it interesting to draw a systematic comparison among these approaches (cf. Table 2.7). First of all, obliviousness is normally defined against the server, but in a multi-client setting it is important to consider it against the clients too (**MC**), since they might be curious or, even worse, collude with the server. This latter aspect is important, since depending on the application, the cloud administrator might create fake clients or just have common interests with one of the legitimate clients. Some constructions allow multiple data owners to operate on the same ORAM (**MD**), while others require them to use disjoint ORAMs: the latter are much less efficient, since if the client does not want to reveal the owner of the accessed entry (e.g., to protect her anonymity, think for instance of the doctor accessing the patient's record), then the client has to perform a fake access to each other ORAM, thereby introducing a multiplicative factor of $O(m)$, where m is the number of data owners. Some constructions require the data owner to periodically access the dataset in order to validate previous accesses (**PI**), some others rely on server-side client synchronization, which can be achieved for instance by a shared log on the server, a gossiping protocol among clients, etc. (**CS**), while others assume a trusted proxy (**Pr**). Among these, gossiping is the mildest assumption since it can be realized directly on the server side as described by [118]. Another aspect to consider is the possibility for the data owner to specify fine-grained access control mechanisms (**AC**). Finally, some constructions enable concurrent accesses to the ORAM (**P**). The final three columns compare the asymptotic complexity of server-side and client-side computations as well as communication.

Franz *et al.* pioneered the line of work on multi-client ORAM, introducing the concept of delegated ORAM [81]. The idea of this construction, based on simple symmetric cryptography, is to let clients commit their changes to the server and to let the data owner periodically validate them according to the access control policy, finally transferring the valid entries into the actual database. Assuming periodic accesses from the data owner, however, constrains the applicability of this technique. Furthermore, this construction does not support multiple data owners. Finally, it guarantees the obliviousness of access patterns with respect to the server as well as malicious clients, excluding however the accesses on data readable by the adversary. While excluding write operations is necessary (an adversary can clearly notice that the data has changed), excluding read operations is in principle not necessary and limits the applicability of the obliviousness definition: for instance, we would like to hide the fact that an oncologist accessed the PHR of a certain patient even from parties with read access to the PHR (e.g., the pharmacy, which can read the prescription but not the diagnosis).

Another line of work, summarized in the lower part of Table 2.7, focuses on the parallelization of client accesses, which is crucial to scale to a large number of clients, while retaining obliviousness guarantees. Most of them [29, 136, 170, 184] assume a trusted proxy performing accesses on behalf of users, with TaoStore [170] being the most efficient and secure among them. These constructions do not formally consider obliviousness against malicious clients nor access control, although a contribution of this work is to prove that a simple variant of TaoStore [170] guarantees both. Finally, instead of a trusted proxy, BCP-OPRAM [37] and CLT-OPRAM [50] rely on a gossiping protocol while PrivateFS [193] assumes a client-maintained log on the server-side, but they do not achieve obliviousness against malicious clients nor access control. Moreover, PrivateFS guarantees concurrent client accesses only if the underlying ORAM already does so.

Other Multi-Client Approaches. Huang and Goldberg have recently presented a protocol for outsourced private information retrieval [103], which is obtained by layering a private information retrieval (PIR) scheme on top of an ORAM data layout. This solution is efficient and conceals client accesses from the data owner, but it does not give clients the possibility to update data. Moreover, it assumes ℓ non-colluding servers, which is due to the usage of information theoretic multi-server PIR.

De Capitani di Vimercati *et al.* [67] proposed a storage service that uses selective encryption as a means for providing fine-grained access control. The focus of their work is to study how indexing data in the storage can leak information to clients that are not allowed to access these data, although they are allowed to know the indices. The authors do, however, neither consider verifiability nor obliviousness, which distinguishes their storage service from ours.

Verifiable outsourced storage. Verifying the integrity of data outsourced to an untrusted server is a research problem that has recently received increasing attention in the literature. Schröder and Schröder introduced the concept of verifiable data streaming (VDS) and an efficient cryptographic realization thereof [174,175]. In a verifiable data streaming protocol, a computationally limited client streams a long string to the server, who stores the string in its database in a publicly verifiable manner. The client has also the ability to retrieve and update any element in the database. Papamathou *et al.* [154] proposed a technique, called streaming authenticated data structures, that allows the client to delegate certain computations over streamed data to an untrusted server and to verify their correctness. Other related approaches are proofs-of-retrievability [177]– [186], which allow the server to prove to the client that it is actually storing all of the client's data, verifiable databases [27], which differ from the previous ones in that the size of the database is fixed during the setup phase, and dynamic provable data possession [77]. All the above do not consider the privacy of outsourced data. While some of the latest work has focused on guaranteeing the confidentiality of the data [189], to the best of our knowledge no existing paper in this line of research takes into account obliviousness.

Personal Health Records. Security and privacy concerns seem to be one of the major obstacles towards the adoption of cloud-based PHRs [43,64,194]. Different cloud architectures have been proposed [135], as well as database constructions [121,132], in order to overcome such concerns. However, none of these works takes into account the threat of a curious storage provider and, in particular, none of them enforces the obliviousness of data accesses.

Part II
Secure Querying of Outsourced Databases

LORD, WHAT CAN THE HARVEST HOPE FOR, IF NOT FOR THE CARE OF THE REAPER MAN?

Terry Pratchett, Reaper Man

3. On the Security of Frequency-Hiding Order-Preserving Encryption

Order-preserving encryption (OPE) is an encryption scheme with the property that the ordering of the plaintexts carry over to the ciphertexts. This primitive is particularly useful in the setting of encrypted databases because it enables efficient range queries over encrypted data. Given its practicality and usefulness in the design of databases on encrypted data, OPE's popularity is growing. Unfortunately, nearly all computationally efficient OPE constructions are vulnerable against ciphertext frequency-leakage, which allows for inferring the underlying plaintext frequency. To overcome this weakness, Kerschbaum recently proposed a security model, designed a frequency-hiding OPE scheme, and analyzed its security in the programmable random oracle model (CCS 2015).

In this work, we demonstrate that Kerschbaum's definition is imprecise and using its natural interpretation, we describe an attack against his scheme. We generalize our attack and show that his definition is, in fact, not satisfiable. The basic idea of our impossibility result is to show that any scheme satisfying his security notion is also IND-CPA-secure, which contradicts the very nature of OPE. As a consequence, no such scheme can exist. To complete the picture, we rule out the imprecision in the security definition and show that a slight adaption of Kerschbaum's tree-based scheme fulfills it.

3.1. Introduction

Outsourcing databases is common practice in today's businesses. The reasons for that are manifold, varying from the sharing of data among different offices of the same company to saving on know-how and costs that would be necessary to maintain such systems locally. Outsourcing information, however, raises privacy concerns with respect to the service provider hosting the data. A first step towards a privacy-preserving solution is to outsource encrypted data and to let the database application operate on ciphertexts. However, simply encrypting all entries does in general not work because several standard queries on the database do no longer work. To maintain as much functionality of the database as possible while adding confidentiality properties, researchers weakened the security properties of encryption schemes to find a useful middle ground. Examples include encryption schemes that support plaintext equality checks, or order-preserving encryption. In this work, we re-visit the recent work on frequency-hiding order preserving encryption by Kerschbaum [115].

Background and related work. Order-preserving encryption (OPE) [34, 179] is arguably the most popular building block for databases on encrypted data, since it allows for inferring the order of plaintexts by just looking at the respective ciphertexts. More precisely, for any two plaintexts p_1 and p_2, whenever $p_1 < p_2$, we have that $\mathcal{E}(p_1) < \mathcal{E}(p_2)$. Hence, OPE allows for efficient range queries and keyword search on the encrypted data. The popularity of this scheme is vouched for by plenty of industrial products (e.g., Ciphercloud[1], Perspecsys[2], and Skyhigh Networks[3]) and research that investigates OPE usage in different scenarios [9,26,98,129,162,163]. Despite the growing popularity and usage in practice, OPE security is debatable. The ideal security notion for OPE is called *indistinguishability against ordered chosen plaintext attacks* (IND-OCPA), which intuitively says that two equally ordered plaintext sequences should be indistinguishable under encryption. Boldyreva *et al.* [34] show that stateless OPE cannot achieve IND-OCPA, unless the ciphertext size is exponential in the plaintext size. Consequently, either one has to relax the security notion or to keep a state.

The former approach has been explored in the context of classical OPE [34, 35, 188] as well as a slightly different notion called *order-revealing encryption* (ORE) [36, 52, 131, 167]. ORE is more general than OPE in the sense that comparison on the ciphertexts can happen by computing a comparison function different from "<". Either way, those schemes do not achieve IND-OCPA but target different, weaker security notions, which allow them to quantify the leakage incurred by a scheme or to restrict the attacker's capabilities. For instance, the scheme by Boldyreva *et al.* [34] is known to leak about the first half of the plaintexts and the scheme by Chenette *et al.* [52] leaks the first bit where two encrypted target plaintexts differ. To date, there exist several works that exploit this extra leakage in order to break OPE applied to different data sets such as medical data and census data [73, 95, 96, 149]. For instance, using a technique based on bipartite graphs, Grubbs *et al.* [96] have recently shown how to break the schemes of Boldyreva *et al.* [34, 35], thereby achieving recovery rates of up to 98%. As opposed to earlier work, this technique works even for large plaintext domains such as first names, last names, and even zip codes.

With regards to the latter approach based on stateful OPE schemes, Popa *et al.* [161] introduced a client-server architecture, where the client encrypts plaintexts using a deterministic encryption scheme and maintains a search tree on the server into which it inserts the ciphertexts. The server exploits the search tree when computing queries on encrypted data. This approach requires a significant amount of communication between the client and the server both for encryption and queries. Similarly, but rather reversed, Kerschbaum and Schroepfer [116] present an OPE scheme where the client stores a search tree that maps plaintexts to ciphertexts. The ciphertexts are chosen such that ordering is preserved and then inserted along with the plaintexts in the search tree. The server only learns the ciphertexts. This approach has less communication between client and server but requires the client to keep a state that is linear in the number of encrypted plaintexts. Both of

[1]`http://www.ciphercloud.com/`
[2]`http://perspecsys.com/`
[3]`https://www.skyhighnetworks.com/`

these schemes are provably IND-OCPA-secure.

Even though these schemes achieve the ideal IND-OCPA security notion, Kerschbaum [115] raises general doubts about the security definition of OPE. Aside the leakage that is introduced by many schemes on top of the order information (e.g., leaking half of the plaintext [34] or the first bit where two plaintexts differ [52]), one central problem of OPE is the leakage of the plaintext frequency. It is easy to distinguish the encryption of data collections in which elements occur with different frequencies. For instance, the encryption of the sequences $1, 2, 3, 4$ and $1, 1, 2, 2$ are not necessarily indistinguishable according to the IND-OCPA security definition.

In order to solve the frequency-leakage problem, Kerschbaum has recently strengthened the IND-OCPA definition of OPE so as to further hide the frequency of plaintexts under the encryption, thus making the encryptions of the above two sequences indistinguishable [115] (CCS 2015). To this end, Kerschbaum introduces the notion of *randomized order*, which is a permutation of the sequence $1, \ldots, n$ where n is the length of the challenge plaintext sequence. Such a permutation is called randomized order if, when applied to a plaintext sequence, the resulting plaintext sequence is ordered with respect to "$\leq$". The original IND-OCPA security definition requires that the two challenge plaintext sequences agree on all such common randomized orders, which implies that every pair of corresponding plaintexts in the two sequences occurs with the same frequency. For instance, this does not hold for the above two sequences $1, 2, 3, 4$ and $1, 1, 2, 2$, since the former can only be ordered using the permutation $(1, 2, 3, 4)$ while the latter can be ordered by any of $(1, 2, 3, 4)$, $(1, 2, 4, 3)$, $(2, 1, 3, 4)$, or $(2, 1, 4, 3)$. Kerschbaum's insight to make the definition frequency-hiding is that the existence of one common randomized order should be sufficient in order not to be able to distinguish them. For instance, the above sequences both share the randomized order $(1, 2, 3, 4)$ and should thus be indistinguishable when encrypted. This intuition is captured by the security notion of *indistinguishability against frequency-analyzing ordered chosen plaintext attacks* (IND-FA-OCPA). Besides devising a novel definition, Kerschbaum also presents a cryptographic instantiation of an OPE scheme and analyzes its security with respect to the new definition in the programmable random oracle model.

Despite the seeming improvement added by Kerschbaum's scheme, Grubbs *et al.* [96] show that using auxiliary information, such as the plaintext distribution that is likely to underlie a certain ciphertext collection, this scheme can be broken with significant recovery rates. In contrast to the practical attacks in [96], our work targets the purely theoretic side of frequency-hiding OPE and we do not consider having auxiliary information at disposal.

3.1.1. Our contributions

In this work, we present both negative and positive results for frequency-hiding order-preserving encryption. On the negative side, we observe that the original definition of IND-FA-OCPA is imprecise [115], which leaves room for interpretation. In particular, the security proof for the scheme presented in [115] seems to suggest that the game challenger chooses a randomized order according to which one of the challenge sequences is encrypted. This fact, however, is not reflected in the definition. Hence, according to a natural interpretation of the definition, we show that it is, in fact, not achievable. We develop this impossibility result for the natural interpretation of IND-FA-OCPA step by step. Investigating on Kerschbaum's frequency-hiding scheme [115], we show that it can actually be attacked–without using auxiliary information as done by Grubbs *et al.* [96]–allowing an adversary to win the IND-FA-OCPA game with very high probability. We further observe that this concrete attack can be generalized into a result that allows us to precisely quantify an attacker's advantage in winning the security game for two arbitrary plaintext sequences that adhere to the security game restrictions. Since Kerschbaum provides formal security claims for his construction [115], we identify where the security proof is incorrect. All these considerations on the concrete scheme finally lead to our main negative result: IND-FA-OCPA security is impossible to achieve or, more precisely, any IND-FA-OCPA secure OPE scheme is also secure with respect to IND-CPA, which clearly contradicts the very functionality of OPE. Hence, such an OPE scheme cannot exist.

As mentioned above, the impossibility of IND-FA-OCPA is bound to an imprecision in the definition in [115], which is only presented informally and lacks necessary information to make it achievable. We hence clarify those imprecisions. The underlying problem of the original definition lies in the capability of the game challenger, which, when reading the definition naturally, is very restricted. The challenger has, for instance, no means to ensure that the encryption algorithm chooses a common randomized order of the two challenge plaintext sequences. To remedy those shortcomings, we devise a more formal definition that removes the consisting imprecisions and makes it possible to devise a frequency-hiding OPE scheme. In particular, we first augment the OPE model, allowing for specifying a concrete ordering when encrypting plaintexts, e.g., to concretely say that the sequence $1, 1, 2, 2$ should be encrypted sticking to the randomized order $(1, 2, 4, 3)$. Secondly, we show that an extension of Kerschbaum's scheme [115], adapted to the new model, is provably secure with respect to the correct definition.

To summarize, our contributions are as follows.

- We show that the original definition of IND-FA-OCPA is imprecise. We then demonstrate that the frequency-hiding OPE scheme of [115] is insecure under a natural interpretation of IND-FA-OCPA. We further generalize the attack, which allows us to rigorously quantify the success probability of an attacker for two arbitrary plaintext sequences. To conclude on the concrete scheme, we identify and explain the problem in the security proof.
- Going one step beyond the concrete scheme, we prove a general impossibility result showing that IND-FA-OCPA cannot be achieved by any OPE scheme.
- We clarify the imprecise points in the original security definition and provide a corrected version called IND-FA-OCPA*.
- To define IND-FA-OCPA* in the first place, we have to augment the OPE model, adding a concrete random order as input to the encryption function.
- Finally, we prove that an extension of [115] fulfills our new definition.

Overall, we believe this work yields a solid foundation for order-preserving encryption, showing that a state-of-the-art security definition is impossible to realize along with an attack on a previously published scheme, and presenting an achievable definition and a concrete realization.

Outline. The rest of the paper is structured as follows. We recall the OPE model and its security definitions in Section 3.2. In Section 3.3 we describe the relevant parts of Kerschbaum's scheme [115]. We present our attack, its generalization, and the problem in the security proof in Section 3.4. Section 3.5 proves the impossibility result. In Section 3.6 we present the augmented OPE model and the definition of IND-FA-OCPA*. Finally, we show that an adaption of [115] to the new model achieves IND-FA-OCPA* in Section 3.7.

3.2. Order-Preserving Encryption

In this section, we briefly review the formal definitions of order-preserving encryption, originally proposed in [179], following the definition adopted in [115].

Definition 3.1 ((Order-Preserving) Encryption)**.** *An* (order-preserving) encryption scheme $\mathcal{OPE} = (\mathsf{K}, \mathcal{E}, \mathcal{D})$ *is a tuple of* PPT *algorithms where*

$S \leftarrow \mathsf{K}(\lambda)$: *The* key generation *algorithm takes as input a security parameter λ and outputs a secret key (or state) S.*

$(S', y) \leftarrow \mathcal{E}(S, x)$: *The* encryption *algorithm takes as input a secret key S and a message x. It outputs a new key S' and a ciphertext y;*

$x \leftarrow \mathcal{D}(S, y)$: *The* decryption *algorithm takes as input a secret key S and a ciphertext y and outputs a message x.*

An OPE scheme is *complete* if for all S, S', x, and y we have that if $(S', y) \leftarrow \mathcal{E}(S, x)$, then $x \leftarrow \mathcal{D}(S', y)$.

The next definition formalizes the property of order preservation for an encryption scheme. Roughly speaking, this property says that the ordering on the plaintext space carries over to the ciphertext space.

Definition 3.2 (Order-Preserving)**.** *An encryption scheme $\mathcal{OPE} = (\mathsf{K}, \mathcal{E}, \mathcal{D})$ is* order-preserving *if for any two ciphertexts y_1 and y_2 with corresponding messages x_1 and x_2 we have that whenever $y_1 < y_2$ then also $x_1 < x_2$.*

This general definition allows for modeling both stateful as well as stateless versions of OPE. We focus on the stateful variant in this paper, hence, the *key S* defined above is actually the state. The definition, moreover, does not specify where the state has to reside, allowing us to model client-server architectures.

3.2.1. Security Definitions

Indistinguishability against ordered chosen plaintext attacks. The standard security definition for order-preserving encryption is indistinguishability against ordered chosen plaintext attacks (IND-OCPA) [34]. Intuitively, an OPE scheme is secure with respect to this definition if for any two equally ordered plaintext sequences, no adversary can tell apart their corresponding ciphertext sequences. IND-OCPA is fulfilled by several schemes (e.g., [116, 161]). We recall the selective version of the definition in the following.

Definition 3.3 (IND-OCPA). *An order-preserving encryption scheme $\mathcal{OPE} = (\mathsf{K}, \mathcal{E}, \mathcal{D})$ has* indistinguishable ciphertexts under ordered chosen plaintext attacks *(IND-OCPA) if* $\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{OCPA}}(\lambda, b) = 1\right]$ *is negligibly (in λ) close to $1/2$ for any* PPT *adversary $\mathcal{A}$ where* $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{OCPA}}(\lambda, b)$ *is the following experiment:*

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{OCPA}}(\lambda, b)$
$(X_0, X_1) \leftarrow \mathcal{A}$
if $|X_0| \neq |X_1| \vee \exists 1 \leq i, j \leq n.\ x_{0,i} < x_{0,j} \oplus x_{1,i} < x_{1,j}$ then
  return $0$
$S_0 \leftarrow \mathsf{K}(\lambda)$
for $1 \leq i \leq |X_0|$ do
  $(S_i, y_{b,i}) \leftarrow \mathcal{E}(S_{i-1}, x_{b,i})$
$b' \leftarrow \mathcal{A}(y_{b,1}, \ldots, y_{b,n})$
if $b = b'$ then
  return $1$
return $0$
```

Definition 3.3 requires that the challenge plaintext sequences are ordered exactly the same, which in particular implies that the plaintext frequency must be the same.

Indistinguishability under frequency-analyzing ordered chosen plaintext attacks. A drawback of the previous definition is that it can be achieved by schemes that leak the plaintext frequency, although any two sequences in which plaintexts occur with different frequencies, e.g., $1, 2, 3, 4$ and $1, 1, 1, 1$, are trivially distinguishable by the attacker. In order to target even such sequences, Kerschbaum [115] proposes a different security definition: instead of requiring the sequences to have exactly the same order, it is sufficient for them to have a common *randomized* order. For a plaintext list X of length n, a randomized order is a permutation of the plaintext indices $1, \ldots, n$ which are ordered according to a sorted version of X. This is best explained by an example: consider the plaintext sequence $X = 1, 5, 3, 8, 3, 8$. A randomized order thereof can be any of $\Gamma_1 = (1, 4, 2, 5, 3, 6)$, $\Gamma_2 = (1, 4, 3, 5, 2, 6)$, $\Gamma_3 = (1, 4, 2, 6, 3, 5)$, or $\Gamma_4 = (1, 4, 3, 6, 2, 5)$, because the order of 3 and 3 as well as the order of 8 and 8 does not matter in a sorted version of X. Formally, a randomized order is defined as follows.

Definition 3.4 (Randomized order)**.** *Let n be the number of not necessarily distinct plaintexts in sequence $X = x_1, x_2, \ldots, x_n$ where $x_i \in \mathbb{N}$ for all i. For a* randomized order *$\Gamma = \gamma_1, \gamma_2, \ldots, \gamma_n$, where $1 \leq \gamma_i \leq n$ and $i \neq j \implies \gamma_i \neq \gamma_j$ for all i, j, of sequence X it holds that*

$$\forall i, j.\ (x_i > x_j \implies \gamma_i > \gamma_j) \ \wedge \ (\gamma_i > \gamma_j \implies x_i \geq x_j)$$

Using this definition, Kerschbaum [115] defines security of OPE against *frequency-analyzing ordered chosen plaintext attacks.* Since the definition is informal in [115], we report the natural way to read the definition.

Definition 3.5 (IND-FA-OCPA)**.** *An order-preserving encryption scheme $\mathcal{OPE} = (\mathsf{K}, \mathcal{E}, \mathcal{D})$ has* indistinguishable ciphertexts under frequency-analyzing ordered chosen plaintext attacks *(IND-FA-OCPA) if $\mathsf{Pr}\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}}(\lambda, b) = 1\right]$ is negligibly (in λ) close to $1/2$ for any* PPT *adversary $\mathcal{A}$ where $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}}(\lambda, b)$ is the following experiment:*

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}}(\lambda, b)$
$(X_0, X_1) \leftarrow \mathcal{A}$
if $|X_0| \neq |X_1| \vee X_0, X_1$ do not have a common randomized order then
  return 0
$S_0 \leftarrow \mathsf{K}(\lambda)$
for $1 \leq i \leq |X_0|$ do
  $(S_i, y_{b,i}) \leftarrow \mathcal{E}(S_{i-1}, x_{b,i})$
$b' \leftarrow \mathcal{A}(y_{b,1}, \ldots, y_{b,n})$
if $b = b'$ then
  return 1
return 0
```

It is clear that while the standard IND-OCPA definition could be achieved, in principle, by a deterministic encryption scheme, the frequency-hiding variant can only be achieved by using randomized ciphertexts since otherwise frequencies are trivially leaked.

Discussion. Comparing the two definitions, we observe that IND-FA-OCPA is a generalization of IND-OCPA since the constraint on the sequences X_0 and X_1 allows for a greater class of instances. In order to see that, we have to consider the constraint, which is

$$\forall 1 \leq i, j \leq n.\ x_{0,i} < x_{0,j} \iff x_{1,i} < x_{1,j}.$$

This constraint is an alternative way of saying that X_0 and X_1 should agree on all randomized orders. Hence, duplicate plaintexts may occur in any of the sequences, but they should occur symmetrically in the other sequence as well.

3.3. Kerschbaum's Construction

We review the OPE scheme of [115]. At a high level, encryption works by inserting plaintexts into a binary search tree that stores duplicates as often as they occur. When an element arrives at its designated node, a ciphertext is selected according to this position.

More formally, let T be a binary tree. We denote by ρ the root of T. For a node $t \in T$ we write $t.m$ to denote the message stored at t and $t.c$ to denote the respective ciphertext. We further use $t.left$ and $t.right$ to denote the left and right child of t, respectively. There are several other parameters: N is the number of distinct plaintexts, n is the number of plaintexts in the sequence that is to be encrypted, $k = \log(N)$ is the required number of bits necessary to represent a plaintext in a node, $\ell = k\lambda$ is this number expanded by a factor of λ and refers to the size of the ciphertexts. Finally, the construction requires a source of randomness, which is called in terms of a function $\mathsf{RandomCoin}()$ (hereafter called $\mathsf{RC}()$ for brevity). According to Kerschbaum, this function can be implemented as a PRF that samples uniformly random bits.

We refer in the following to the client as the one storing the binary tree. This is well motivated in the cloud setting where the client outsources encrypted data to the cloud server who may not have access to the actual message-ciphertext mapping. One may wonder why a client that anyway has to store a mapping of plaintexts to ciphertexts cannot simply store the data itself: Kerschbaum also presents an efficient compression technique for the tree which in some cases can lead to compression ratios of 15.

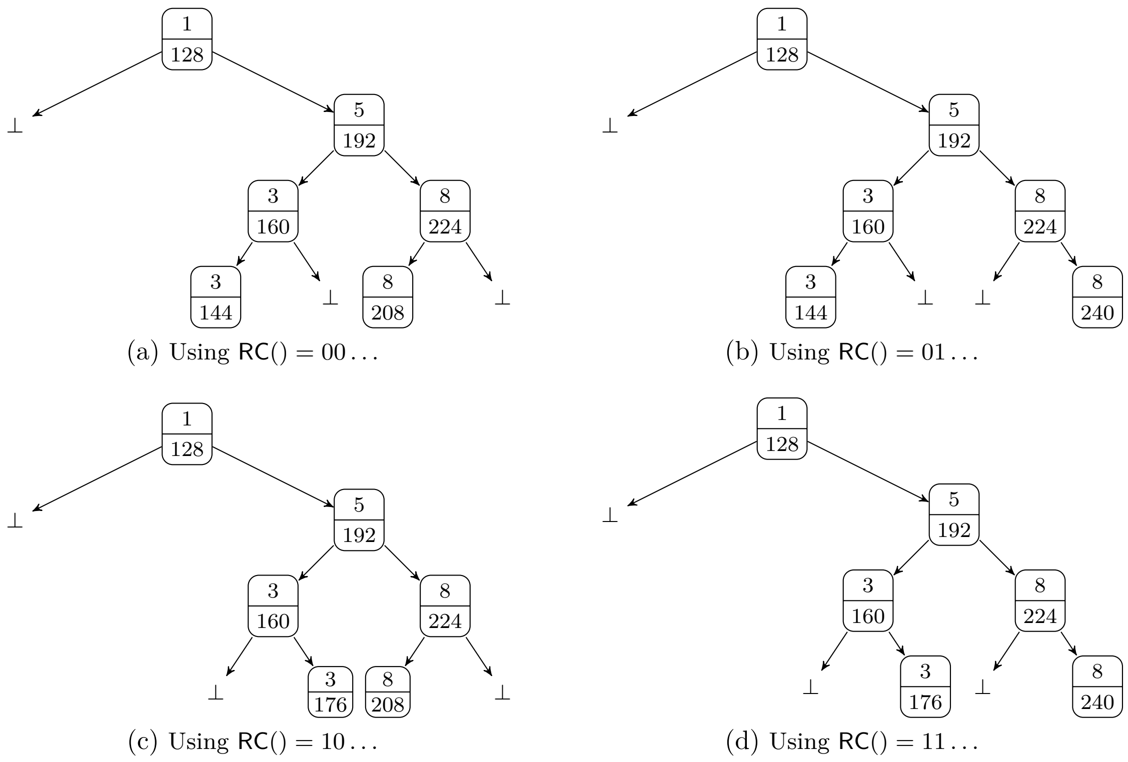
Implementation of $S \leftarrow \mathsf{K}(\lambda)$. The client sets up an empty tree T. The state S consists of the tree T as well as all parameters k, ℓ, n, and N. Furthermore, S contains the minimum ciphertext $min = -1$ and the maximum ciphertext $max = 2^{\lambda \log(n)}$. These minimum and maximum numbers are only necessary to write the encryption procedure in a recursive way. Usually, n is not known upfront, so it has to be estimated. If the estimation is too far from reality, the tree has to be freshly setup with new parameters.

Algorithm 15 $\mathcal{E}(S, x)$ where $S = t, min, max$

```
1:  if $t = \perp$ then
2:      $t.m = x$
3:      $t.c = min + \lfloor \frac{max - min}{2} \rfloor$
4:      if $t.c = 0$ then
5:          rebalance the tree
6:      end if
7:      return $t.c$
8:  end if
9:  $b \leftarrow -1$
10: if $x = t.m$ then
11:     $b \leftarrow \mathsf{RC}()$
12: end if
13: if $b = 1 \vee x > t.m$ then
14:     $\mathcal{E}(t.right, t.c + 1, max, x)$
15: else
16:     if $b = 0 \vee x < t.m$ then
17:         $\mathcal{E}(t.left, min, t.c - 1, x)$
18:     end if
19: end if
```

Implementation of $(S', y) \leftarrow \mathcal{E}(S, x)$. To encrypt a plaintext x, the client proceeds as follows. Whenever the current position in the tree is empty (especially in the beginning when the tree is empty), the client creates a new tree node and inserts x as the plaintext (lines 15.1–15.8). The ciphertext is computed as the mean value of the interval from min to max (line 15.3). In particular, the first ciphertext will be $2^{\lambda \log(n) - 1}$. Whenever there is no ciphertext available (line 15.4), the estimation of n has shown to be wrong and the tree has to be rebalanced. We do not detail this step here since it is not important for our attack; instead we refer the interested reader to [115] for a detailed explanation. If instead, the current position in the tree is already occupied and the message is different from x, then we either recurse left (line 15.17) or right (line 15.14) depending on the relation between the occupying plaintext and the one to be inserted. The same happens in case x is equal to the stored message, but then we use the RC procedure to decide where to recurse (lines 15.9–15.12).

Implementation of $x \leftarrow \mathcal{D}(S, y)$. To decrypt a given ciphertext y, we treat the tree as a binary search tree where the key is $t.c$ and search for y. We return $t.m$ as soon as we reach a node t where $t.c = y$.



(a) Using $\mathsf{RC}() = 00\ldots$ (b) Using $\mathsf{RC}() = 01\ldots$

(c) Using $\mathsf{RC}() = 10\ldots$ (d) Using $\mathsf{RC}() = 11\ldots$

Figure 3.1.: An example for different binary search trees after inserting the sequence $X = 1, 5, 3, 8, 3, 8$, depending on the output of RC.

Example 3.1. To simplify the access to the construction, we review a detailed example. Figure 3.1 shows the four possible resulting binary search trees after inserting $X = 1, 5, 3, 8, 3, 8$, depending on the output of RC. We use a ciphertext space of $\{1, \ldots, 256\}$. Each different output of RC corresponds to one of the four possible randomized orders Γ_i for $1 \leq i \leq 4$.

Security. The scheme is proven secure against frequency-analyzing ordered chosen plaintext attack. To this end, [115] constructs a simulator which, given the two challenge plaintext sequences, produces identical views independent of which of two sequences is chosen. We investigate on the proof in the next section.

3.4. An Attack on Kerschbaum's FH-OPE Scheme

In this section, we investigate on the security achieved by Kerschbaum's construction [115]. In order to start, we observe that Kerschbaum proves his construction secure. However, as we show later in this section, the security proof makes an extra assumption on the game challenger's capabilities, namely that the challenger can dictate the randomized order used by the encryption algorithm to encrypt either challenge plaintext sequence. Using the natural interpretation of IND-FA-OCPA (see Definition 3.5), this additional assumption is not justified and, hence, Kerschbaum's scheme is no longer secure. We thus present a concrete attack, which is related to the distribution based on which randomized sequences are chosen for encryption (Section 3.4.1). We then explain more in detail why Kerschbaum's security result is incorrect with respect to Definition 3.5 (Section 3.4.2). Finally, we show that even if randomized orders are chosen uniformly at random, the scheme is still vulnerable (Section 3.4.3).

3.4.1. A Simple Attack

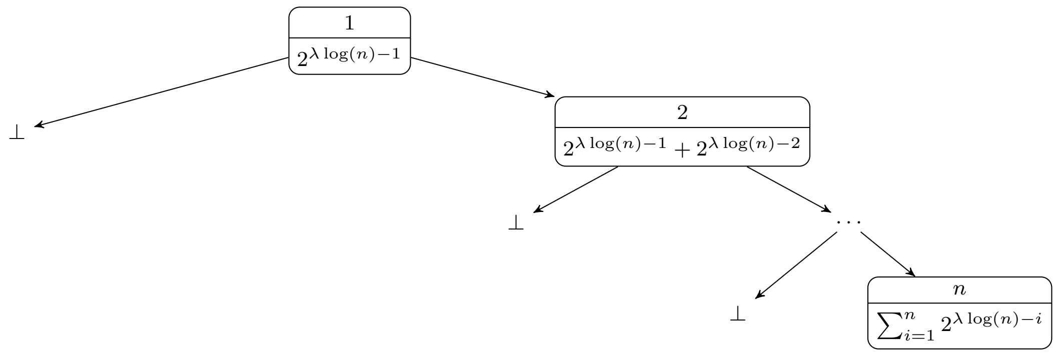

1
$2^{\lambda \log(n)-1}$
$\perp$
2
$2^{\lambda \log(n)-1} + 2^{\lambda \log(n)-2}$
$\perp$
$\cdots$
$\perp$
n
$\sum_{i=1}^{n} 2^{\lambda \log(n)-i}$

Figure 3.2.: The resulting binary search tree when encrypting sequence X_0.

Our attack consists of two plaintext sequences that are given to the challenger of the FA-OCPA game, who encrypts step-by-step randomly one of the two sequences. By observing the sequence of resulting ciphertexts, we will be able to determine which sequence the challenger chose with very high probability.

Consider the two plaintext sequences $X_0 = 1, 2, 3, \ldots, n$ and $X_1 = 1, \ldots, 1$ such that $|X_0| = |X_1| = n$. Clearly, both X_0 and X_1 have a common randomized order, namely $\Gamma = 1, 2, 3, \ldots, n$, i.e., the identity function applied to X_0. Moreover, consider the binary search tree produced by the scheme when encrypting X_0 in Figure 3.2. This tree is generated fully deterministically since the elements in X_0 are pairwise distinct, or equivalently, X_0 has only a single randomized order. Now let us investigate how X_1 would be encrypted by

Algorithm 15. In every step, the RC procedure has to be called in order to decide where to insert the incoming element. If coins are drawn uniformly at random then only with a probability of $1/2^{n(n-1)/2}$ RC will produce the bit sequence $1 \dots 1$ of length $n(n-1)/2$, which is required in order to produce the exact same tree as in Figure 3.2. Notice that the RC sequence must be of length $n(n-1)/2$ since for every plaintext that is inserted on the right, the function has to be called once more. Hence, the Gaussian sum $\sum_{i=1}^{n-1} i$ describes the number of required bits. Consequently, an adversary challenging the FH-OCPA challenger with X_0 and X_1 will win the game with probability $(1/2)(1-1/2^{n(n-1)/2})$ where the factor $1/2$ accounts for the probability that the challenger chooses X_1. Hence, if X_1 is chosen, our attacker wins with overwhelming probability, otherwise he has to guess. Notice that the combined probability is nevertheless non-negligible.

In conclusion, the central observation is that the number of calls to RC strongly depends on how many equal elements are met on the way to the final destination when encrypting an element. Therefore, not all ciphertext trees are equally likely.

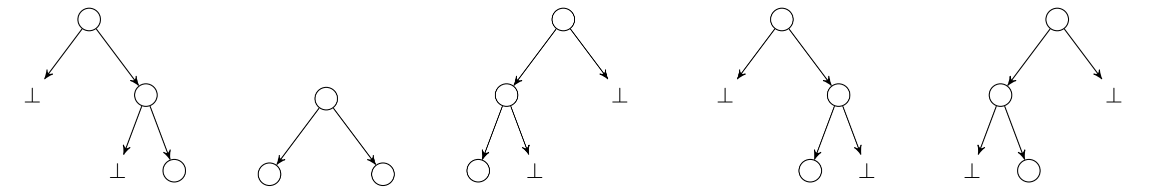
3.4.2. Understanding the Problem

In this section, we analyze the core of the problem.

Artifacts of the Construction. The analysis in the previous section shows that randomized orders are not drawn uniformly random. Otherwise, the adversary's success probability would be $(1/2)(1-1/n!)$ since X_1 has $n!$ many randomized orders and the probability that a specific one is chosen uniformly is $1/n!$. Instead, we analyzed the probability that the *encryption algorithm* chooses that specific randomized order, which depends on the number of calls to RC and its results, which should all be 1.

In order to exemplify this artifact, we consider the sequence $1, 1, 1$. We depict the different trees when encrypting the sequence in Figure 3.3. As we can see, different trees require a different number of calls to RC, and have thus a different probability of being the encryption tree. On the one hand, the trees in Figure 3.3a and Figure 3.3c–3.3e all have a probability of $1/8$ to be the result of the encryption since each of them requires RC to be called three times. On the other hand, the two trees in Figure 3.3b have a probability of $1/4$ of being chosen each since RC has to be called only twice.

To formally capture the probability range of different randomized orders, we want to understand which randomized orders are most probable and which ones are least probable. Before we start the analysis, we observe that it does not matter whether we consider the probability or the number of calls to RC, since every call to RC adds a factor of $1/2$ to the probability that a certain randomized order is chosen. So as we have seen in the concrete counter-example in the previous section and the example above, a tree with linear depth represents the least likely randomized order since it requires the most calls to RC, which increases by one every time a new element is encrypted. Conversely, randomized orders represented by a perfectly balanced binary tree are more likely since they require the minimum number of calls to RC. Let H be the histogram of plaintext occurrences in a sequence. Then, as before, the number of calls to RC can be computed as the sum over



(a) $\Gamma_1 = (1,2,3)$, $\mathsf{RC}() = 111$.

(b) $\Gamma_2 = (2,1,3)$, $\mathsf{RC}() = 01$, and $\Gamma_3 = (2,3,1)$, $\mathsf{RC}() = 10$.

(c) $\Gamma_4 = (3,2,1)$, $\mathsf{RC}() = 000$.

(d) $\Gamma_5 = (1,3,2)$, $\mathsf{RC}() = 110$.

(e) $\Gamma_6 = (3,1,2)$, $\mathsf{RC}() = 001$.

Figure 3.3.: The trees displaying different randomized orders for the sequence $1, 1, 1$.

every node's depth in the subtree in which all duplicate elements reside, which is at least

$$\begin{aligned}\sum_{p \in X} \frac{\sum_{i=1}^{H(p)} \log(i)}{H(p)} &= \sum_{p \in X} \frac{\log(H(p)!)}{H(p)} \geq \sum_{p \in X} \frac{\frac{H(p)}{2} \log\left(\frac{H(p)}{2}\right)}{H(p)} \\ &= -\frac{|X|}{2} + \frac{1}{2} \sum_{p \in X} \log(H(p))\end{aligned}$$

where we make use of the fact that $\left(\frac{n}{2}\right)^{\frac{n}{2}} \leq n! \leq n^n$ in the first inequality. All other randomized orders lie probability-wise somewhere in between.

Proof Technique. Despite our counter-example, the scheme is proven secure in the programmable random oracle model [115]; this obviously constitutes a contradiction. To understand the problem in depth, we have to have a closer look at the security proof. The idea behind the proof is as follows: the challenger selects uniformly at random a common randomized order with respect to the two challenge sequences. This common randomized order is then given as source of randomness to the random oracle which answers questions accordingly. More precisely, let $\Gamma = (\gamma_1, \dots, \gamma_n)$ be the selected order. Whenever the algorithm cannot decide where to place a plaintext x_j in the search tree, i.e., $x_i = x_j$ for $i < j$, meaning that x_i is an entry that is already encrypted in the tree, then the challenger asks the random oracle for a decision on i and j. The oracle answers with 1 if $\gamma_i < \gamma_j$ and with 0 if $\gamma_i > \gamma_j$ (notice that $\gamma_i \neq \gamma_j$ by Definition 3.4). In this way, the challenger produces a search tree and corresponding ciphertexts that are valid for both challenge sequences and which are in fact independent of the bit. Hence, the adversary has no chance of determining which sequence has been chosen other than guessing.

The Simulation is Incorrect. The proof strategy clearly excludes the attack described in the previous section. The reason is that the RC function is supposed to output *uniformly random* coins. As we have seen, even if RC outputs truly random coins then not every possible randomized order of the chosen sequence is equally likely. Hence, the choice of the random sequence is in two aspects unfaithful: first, the challenger restricts the number of randomized orders to those that both sequences have in common while RC does not know the two sequences and can, hence, not choose according to this requirement. Second, the fact that the choice is uniform does not reflect the reality. As we have seen, one artifact of the construction is that not every randomized order is equally likely. Consequently, forcing RC to generate output based on a common randomized order changes the distribution from which coins are drawn and, hence, neither all randomized orders are possible nor are their probabilities of being chosen correctly distributed. In the extreme case described in Section 3.4.1, it even would disallow all but one randomized order to be the result of the encryption routine. As a consequence, the proof technique changes the behavior of RC to an extent that makes the simulation incorrect.

3.4.3. Generalizing the Attack in an Ideal Setting

Since the scheme is vulnerable to an attack and it chooses the randomized order under which a sequence is encrypted in a non-uniform way, we find it interesting to also investigate whether the scheme is still vulnerable in an ideal setting where the choice of the randomized order happens uniformly.

The answer to this question is unfortunately positive, as the following result shows. Concretely, only if two sequences agree on essentially all randomized orders, the adversary has a negligible advantage of distinguishing them.

Theorem 3.1. *Let X_0 and X_1 be two plaintext sequences of length n. Further assume that X_0 has m_0 and X_1 has m_1 randomized orders, respectively, and that they have m randomized orders in common. Then, for the idealized construction of [115] which encrypts plaintexts under a uniformly chosen randomized order, there exists an adversary whose success probability in winning the IND-FA-OCPA game is at least $1 - m\frac{m_0+m_1}{2m_0m_1}$.*

Proof. We construct an adversary, which submits both X_0 and X_1 to the FA-OCPA challenger. Since X_0 has m_0 randomized orders, the probability that one of those in common with X_1 is chosen by the encryption procedure is $\frac{m}{m_0}$ due to the uniformly random behavior. Likewise, for X_1, the probability that a common randomized order is chosen by the encryption procedure is $\frac{m}{m_1}$. Hence, depending on the challenger's bit b, the adversary sees a non-common randomized order with probability $1 - \frac{m}{m_b}$, which also reflects its success probability for winning the game when the challenger picks b. Consequently,

$$\begin{aligned}\mathsf{Pr}\left[\mathcal{A} \text{ wins}\right] &= \left|\mathsf{Pr}\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}}(\lambda, 1) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}}(\lambda, 0) = 1\right]\right| \\ &= \frac{1}{2}\left(1 - \frac{m}{m_0}\right) + \frac{1}{2}\left(1 - \frac{m}{m_1}\right) = 1 - m\frac{m_0 + m_1}{2m_0m_1}\end{aligned}$$

□

In the example from the previous section, we have parameters $m_0 = 1$, $m_1 = n!$, and $m = 1$. Substituting those into Theorem 3.1, we get the aforementioned non-negligible success probability of

$$1 - m\frac{m_0 + m_1}{2m_0m_1} = 1 - \frac{1+n!}{2n!} = \frac{1}{2}\left(1 - \frac{1}{n!}\right).$$

3.5. Impossibility of IND-FA-OCPA

The previously presented results raise the question if IND-FA-OCPA, as presented in [115] can be achieved at all. It turns out that this is not the case: in this section, we prove an impossibility result for frequency-hiding order-preserving encryption as defined in Definition 3.5. Formally we prove the following theorem.

Theorem 3.2. *Let X_0 and X_1 be two arbitrary plaintext sequences of the same length that do not share any randomized order. Let furthermore $\mathcal{OPE}$ be an order-preserving encryption scheme secure against IND-FA-OCPA. Then, the probability of distinguishing whether X_0 is encrypted or whether X_1 is encrypted with $\mathcal{OPE}$ is negligibly close to 1/2.*

Before we prove the theorem using Definition 3.5, we argue why it implies the impossibility of frequency-hiding OPE. According to the theorem, no adversary can distinguish the encryptions of two arbitrary sequences of his choice that are ordered in a completely different manner. This constitutes a formulation of the IND-CPA property for multiple messages, restricted to sequences that do not share a randomized order. The restriction, however, is not necessary since sequences that share a randomized order are trivially indistinguishable by IND-FA-OCPA. To exemplify, let the two sequences be $X_0 = 1, 2, \ldots, n$ and $X_1 = n, n-1, \ldots, 1$, so X_1 is the reverse of X_0. According to Theorem 3.2, no adversary can distinguish which one of the two is encrypted. However, due to the correctness of OPE it must be the case that the encryption Y^* fulfills for all i and j and $b \in \{0, 1\}$

$$y_i^* \geq y_j^* \implies x_i^b \geq x_j^b.$$

Consequently, if X_0 is encrypted we have $y_i^* < y_j^*$ for $i < j$ and vice versa for X_1. Hence, an adversary could trivially distinguish which of the two sequences is encrypted. Hence, by contraposition of Theorem 3.2, an IND-FA-OCPA-secure OPE scheme cannot exist.

Proof of Theorem 3.2. In the security game $G(b)$, on input X_0 and X_1 by $\mathcal{A}$, the challenger encrypts sequence X_b, gives the ciphertexts Y^* to $\mathcal{A}$, who replies with a guess b'. $\mathcal{A}$ wins if $b = b'$.

We define three games. $G_1 = G(0)$. For G_2, we select a sequence X^* which has a randomized order in common with both X_0 and X_1. Notice that such a sequence always exists, e.g., take the series $a, a, \ldots, a$ (n times) for arbitrary a in an appropriate domain. Instead of encrypting X_0 as in G_1, we now encrypt X^*. Finally, $G_3 = G(1)$.

In order to show that $G_1 \approx G_2$, assume that there exists a distinguisher $\mathcal{A}$ that can distinguish between G_1 and G_2 with non-negligible probability. Then we construct a reduction $\mathcal{B}$ that breaks IND-FA-OCPA. On $\mathcal{A}$'s input, $\mathcal{B}$ forwards X_0 and a sequence X^* to the IND-FA-OCPA challenger. The challenger answers with Y^*, which $\mathcal{B}$ again forwards to $\mathcal{A}$. $\mathcal{A}$ outputs a bit b', which $\mathcal{B}$ again forwards to the challenger. The simulation is obviously efficient. If the internal bit of the IND-FA-OCPA challenger is 0, we perfectly simulate G_1, while we simulate G_2 when the bit is 1. Hence, the success probability of $\mathcal{A}$ carries over to $\mathcal{B}$ since $\mathcal{B}$ only forwards messages. Since we assumed that $\mathcal{A}$ can successfully distinguish G_1 and G_2 with non-negligible probability, it must be the case that $\mathcal{B}$ wins the IND-FA-OCPA game with non-negligible probability. This is a contradiction.

The proof of $G_2 \approx G_3$ is symmetric to the one above. In conclusion, we have that $G_1 \approx G_2 \approx G_3$, and hence, $G(0) \approx G(1)$, meaning that every adversary can distinguish between encryptions of X_0 and X_1 that do not share a randomized order only with negligible probability. $\square$

3.6. A New Definition for FH-OPE

Since the notion of indistinguishable ciphertexts under frequency-analyzing ordered chosen plaintext attacks is not achievable, it is desirable to understand the problem of the original definition and try to come up with a suitable one that still captures the idea of frequency-hiding but is achievable.

Interestingly enough, the solution to our problem can be found by investigating again Kerschbaum's security proof. The proof builds a random oracle that overcomes the issues of the definition. Even though this construction of the oracle is incorrect, as we have shown previously, it helps us identify the problem with the definition. In the definition, the challenger has no means to tell the encryption algorithm which randomized order to choose when encrypting the chosen challenge sequence. Hence, it could be the case that the algorithm chooses an order that is not common to both challenge sequences. Had the challenger a way to decide which order to encrypt with, the problem were gone.

Consequently, we tackle the problem from two angles: (1) we augment the OPE model by one more input to the encryption function, namely, the randomized order that is supposed to be used and (2) we strengthen the challenger's capabilities during the security game: it may now, additionally to selecting which sequence to encrypt, also choose a common randomized order as input to the encryption algorithm. This new definition still captures the notion of frequency-hiding in the same way, it just excludes the attacks presented in this work and makes the definition, thus, achievable.

3.6.1. Augmented OPE Model

We present the augmented model in the following definition. Notice that the only difference to Definition 3.1 is the additional input Γ to the encryption function. This additional input serves the purpose of deciding from outside of the function, which randomized order should be used to encrypt the plaintexts. In contrast, standard OPE decides about the ordering randomly inside of the function. We stress that augmented OPE is more general than OPE since the input Γ can be replaced by the result of a call to a random function.

Definition 3.6 (Augmented OPE). *An* augmented order-preserving encryption scheme $\mathcal{OPE}^* = (\mathsf{K}, \mathcal{E}, \mathcal{D})$ *is a tuple of* PPT *algorithms where*

$S \leftarrow \mathsf{K}(\lambda)$: *The* key generation *algorithm takes as input a security parameter λ and outputs a secret key (or state) S;*

$(S', y) \leftarrow \mathcal{E}(S, x, \Gamma)$: *The* encryption *algorithm takes as input a secret key S, a message x, and an order Γ and outputs a new key S' and a ciphertext y;*

$x \leftarrow \mathcal{D}(S, y)$: *The* decryption *algorithm $x \leftarrow \mathcal{D}(S, y)$ takes as input a secret key S and a ciphertext y and outputs a message x.*

3.6.2. The New Definition IND-FA-OCPA*

The new security game is close in spirit to Definition 3.5. The difference is that (1) it is defined over an augmented OPE scheme which makes the randomized order used for encryption explicit and (2) the challenger chooses that order uniformly at random from the orders that both challenge sequences have in common. Since we define the notion adaptively, we introduce some new notation with respect to randomized orders.

In the following definition, we let $\Gamma = \gamma_1, \dots, \gamma_n$ and we use the notation $\Gamma{\downarrow_i}$ to denote the order of the sequence $\gamma_1, \dots, \gamma_i$. Notice that this order is unique since Γ is already an order. For instance, take the randomized sequence $\Gamma = 1, 6, 4, 3, 2, 5$. Then, $\Gamma{\downarrow_3} = 1, 3, 2$, which is the order of $1, 6, 4$.

Definition 3.7 (IND-FA-OCPA*). *An augmented order-preserving encryption scheme* $\mathcal{OPE}^* = (\mathsf{K}, \mathcal{E}, \mathcal{D})$ *has* indistinguishable ciphertexts under frequency-analyzing ordered chosen plaintext attacks *if* $\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}^*}(\lambda, b) = 1\right]$ *is negligibly (in λ) close to $1/2$ for any* PPT *adversary $\mathcal{A}$ where $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}^*}(\lambda, b)$ is the following experiment:*

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{FA-OCPA}^*}(\lambda, b)$
$(X_0, X_1) \leftarrow \mathcal{A}$
if $|X_0| \neq |X_1| \vee X_0, X_1$ do not share a common randomized order then
  return 0
Select $\Gamma$ uniformly at random from the common randomized orders of $X_0, X_1$
$S_0 \leftarrow \mathsf{K}(\lambda)$
for $1 \leq i \leq |X_0|$ do
  $(S_i, y_{b,i}) \leftarrow \mathcal{E}(S_{i-1}, x_{b,i}, \Gamma{\downarrow_i})$
$b' \leftarrow \mathcal{A}(y_{b,1}, \dots, y_{b,n})$
if $b = b'$ then
  return 1
return 0
```

3.7. Constructing Augmented OPE

We show how to construct an augmented OPE scheme. Interestingly enough, the key observation to $\mathcal{OPE}^*$ is that the scheme of [115], which we presented in Section 3.3 can be modified so as to fit the new model.

As we introduce a third input to the encryption function, namely an order that is as long as the currently encrypted sequence plus one, we have to show how to cope with this new input in the construction. The key idea is quite simple: usually, the encryption scheme draws randomness from a PRF when the plaintext to be encrypted is already encrypted, in order to decide whether to move left or right further in the tree. The additional input solves this decision problem upfront, so there is no need for using randomness during the encryption.

While the setup and re-balancing algorithms are as described in [115], we describe the new encryption algorithm in Algorithm 16. Furthermore, we require that every node in the tree stores its index in the plaintext sequence, i.e., we add an attribute index to each node t. We further assume that the index of the message that is currently to be encrypted is the length of the order Γ. As we can see, the only difference between Algorithm 15 and Algorithm 16 is the behavior when the message to be inserted is equal to the message currently stored at t. Then, the order Γ is considered so as to decide whether to traverse the tree further to the left or right.

Algorithm 16 $\mathcal{E}(S, x, \Gamma)$ where $S = t, min, max$ and $\Gamma = \gamma_1, \ldots, \gamma_k$

```
1:  if $t = \perp$ then
2:      $t.m = x$
3:      $t.\mathsf{index} = k$
4:      $t.c = min + \lfloor \frac{max - min}{2} \rfloor$
5:      if $t.c = 0$ then
6:          rebalance the tree
7:      end if
8:      return $t.c$
9:  end if
10: $b \leftarrow -1$
11: if $x = t.m$ then
12:     $b \leftarrow \gamma_k > \gamma_{t.\mathsf{index}}$
13: end if
14: if $b = 1 \vee x > t.m$ then
15:     $\mathcal{E}(t.right, t.c + 1, max, x, \Gamma)$
16: else
17:     if $b = 0 \vee x < t.m$ then
18:         $\mathcal{E}(t.left, min, t.c - 1, x, \Gamma)$
19:     end if
20: end if
```

The encryption algorithm also nicely demonstrates that it does not matter from which domain the ordering draws its elements. The only important property of such an ordering is the relation of the single elements to each other, i.e., that (1) all elements of the order are distinct and (2) whether $\gamma_i < \gamma_j$ or the other way around. We do, hence, not require the shrinking function $\Gamma{\downarrow_i}$ for this construction: when encrypting an overall sequence of plaintexts with a predetermined randomized order Γ, it is sufficient to just cut the Γ to size i when encrypting the i-th element. The reason is that the relative ordering of the first i elements is not changed after shrinking, which suffices to let the algorithm decide about where to branch.

3.7.1. Formal Guarantees

We argue in this section that the tree-based scheme presented in the previous section is IND-FA-OCPA* secure. The reason for that is as follows: first, the challenger can select a randomized order that is in common to both sequences and give that chosen order to the encryption algorithm. Second, no matter which of the two sequences is encrypted according to the order, the resulting ciphertexts are equivalent in both cases. Hence, the adversary cannot do better than guessing the bit, since the ciphertexts are independent of the underlying plaintexts.

Theorem 3.3. *The $\mathcal{OPE}^*$ scheme presented in Section 3.7 is* IND-FA-OCPA* *secure.*

Proof. Let $\mathcal{A}$ be an arbitrary adversary for the game in Definition 3.7. Let furthermore X_0 and X_1 be the two plaintext sequences chosen by $\mathcal{A}$. By definition, those sequences share at least one common randomized order. Let Γ be one of those common orders, selected uniformly at random from the universe of common randomized orders. When encrypting either X_0 or X_1, Algorithm 16 uses Γ to decide where to branch. Hence, Algorithm 16's decisions are independent of the input plaintext sequence, and thus independent of the chosen bit b. Consequently, all information that $\mathcal{A}$ receives from the challenger are independent of b and he can thus, only guess what b is. This concludes the proof. □

Part III
Coercion-Resistant Rating Platforms

You can't go around building a better world for people. Only people can build a better world for people. Otherwise it's just a cage.

Terry Pratchett, Witches Abroad

4. CRRP: Coercion-Resistant Rating Platforms

User ratings are a central asset for a variety of online services and an essential feature is their trustworthiness, which can be reduced to four fundamental properties: *authentication* of users in order to avoid double-rating, *public verifiability* of the rating procedure in order to prevent compromised service providers from biasing the results, *privacy* of users and their ratings, which is necessary in sensitive scenarios, and *coercion-resistance*, which is crucial to avoid rate buying or forced rating. While the first three properties can be achieved by state-of-the-art privacy-enhancing technologies such as attribute-based credentials (ABCs), realizing a rating platform that achieves the four of them is an open problem. While coercion-resistance has been studied in remote e-voting protocols, achieving this property in the presence of ABCs calls for profound technical innovations, since existing e-voting solutions for coercion-resistance are incompatible with ABCs.

In this chapter, we present CR^{2}P, the first ABC-based coercion-resistant rating platform. Our cryptographic construction relies on a novel ABC construction called signatures on randomizable commitments, which hides any sensitive user attribute, even from the user herself in order to protect her privacy in case of coercion, and a coercion-resistant user identification token, which determines the owner of an ABC. We formally defined and proved the security properties offered by CR^{2}P, and we conducted an experimental evaluation demonstrating the feasibility of our approach. Finally, we show that CR^{2}P can be also used for remote e-voting, featuring a stronger attacker model than state-of-the-art coercion-resistant remote e-voting protocols.

4.1. Introduction

Building trust in rating platforms. Rating platforms play a crucial role in a number of online services, such as e-healthcare, travel agencies, and e-commerce. A fundamental requirement for a functioning rating platform is its *trustworthiness*, which can be characterized by four fundamental properties, namely, authentication, privacy, verifiability, and coercion-resistance, as discussed below.

Authentication of users is necessary to avoid double-rating or rating from unauthorized users. This is typically enforced by letting users login into the system via their username and password. This habit, however, clearly gives rise to a *privacy* problem, since users might want to hide their identity in sensitive scenarios (e.g., e-healthcare) or fear to be punished in case their vote is leaked by the service provider (e.g., a student negatively rating a lecture on the university's rating course evaluation platform). Another crucial property is the *verifiability* of the rating procedure, which is crucial in case users do not necessarily trust the service provider (e.g., a university might have an interest in demonstrating a large appreciation of its courses, or an e-health rating platform might be financed by doctors themselves). These three properties, despite their seemingly contradictory nature, can be achieved by so called attribute-based credentials [14, 18, 23, 24, 39, 74, 84, 105, 143]: the idea is to let the service provider give users a digital signature on their attributes (e.g., student id, registered courses, etc.) and to let them authenticate through zero-knowledge proofs revealing the properties of the attributes required for authentication and nothing else (e.g., for a course evaluation, the fact that the student is registered in the current edition of that course, or for an e-health platform, the identity of the doctor performing the diagnosis). Attribute-based credentials further support so called service-specific pseudonyms [143], a pseudonym mechanism which allows one to track user actions within a certain service (e.g., to check that each patient rates the same doctor at most once) while making user actions unlinkable across different services (e.g., the ratings of different doctors in order to prevent one from recovering the identity of the patient by correlating the doctors she has been treated by).

A final, crucial property of rating platforms is the resistance to coercion, which can take the form of rate buying [7, 61, 79, 156] and forced rating [20, 164]. Intuitively one would like to prevent a coercer forcing the user to give away her secret material to rate: attribute-based credentials, unfortunately, fall short of providing coercion-resistance [120], since the mere possession of the credential suffices to rate. Furthermore, the credential itself reveals all the user attributes, thereby giving up any form of attribute confidentiality in case of coercion.

Coercion-resistance in e-voting. Coercion-resistance is well studied in the e-voting literature [112], where it is known to be hard to achieve and indeed provided only by a handful of protocols. A common technique [112] to achieve coercion-resistance is to let the registrar (the credential issuer) equip the voter with a credential that allows her to form a ballot, which can be verified by the talliers (the service provider). In case of coercion, the credential and all transcripts related to it can be faked by the voter, leaving the coercer oblivious to whether the credential is real or not. A sophisticated tallying protocol, typically based on mixing, allows for dropping invalid votes without revealing to the coercer whether his vote has been counted or not, which is crucial to preserve coercion-resistance. Interestingly enough, the underlying attacker model considers both a distributed registrar and tallier, where at least one party per role must be honest. This attacker model is not compatible with the setup of rating platforms as it is unclear how to distribute the credential issuer, for instance, the doctor in an e-health scenario or the university in a course evaluation system. Furthermore, the tallying procedure used in e-voting protocols typically relies on information provided by the registrar to the service provider, while it is unrealistic, and arguably undesirable given the supposedly independent nature of rating platforms, to assume in the latter a direct communication between credential issuers and service providers.

4.1.1. Our contributions

In this chapter, we present CR^{2}P, the first privacy-preserving rating platform that supports *attribute-based authorization* with *attribute-hiding*, *anonymous ratings*, *coercion-resistance*, and *verifiability.* We provide rigorous cryptographic definitions for all these properties in a strong attacker model, where the credential issuer,[1] the other users, and all service providers but one can arbitrarily misbehave and collude.[2] Following the literature, we assume a distributed service provider handling the rating procedure: in the case of rating platforms, we believe service providers have an incentive in joining efforts and building a trustworthy environment for users to post their ratings, given the importance of ratings in marketing strategies. In particular, the contributions of this chapter can be summarized as follows:

- a definitional framework for rating platforms, capturing *attribute-hiding*, *anonymity*, *coercion-resistance*, and *verifiability* in terms of game-based definitions;

- the first cryptographic construction provably achieving the aforementioned properties. This is based on two novel cryptographic primitives, namely, *signatures on randomizable commitments* (SRC) and coercion-resistant identification tokens. The former, in particular, is a primitive of independent interest, which consists of a signature on randomizable commitments, which does not leak the committed values and allows for a two stage verification. In particular, a user owning a SRC can randomize it and start the verification against a subset of the committed values, thereby creating an intermediate signature, which is unlinkable to the original one and whose verification can be finalized against the remaining committed values (if the opening information is known). In our protocol, this functionality is leveraged as follows: for each of her attributes, the user is provided with a SRC by the credential issuer that combines the attribute with information linkable to the user, e.g., the user's verification key. In order to form a rating, the user randomizes the signature and its commitment for unlinkability reasons and starts the verification against her verification key. She sends the so-created intermediate signature to the rating system along with a proof of knowledge of the matching signing key. The rating system can then finalize the verification against the attributes necessary for authorizing the rating, which are provided in encrypted form and are only revealed in the tallying phase.

- a proof of security against an attacker model that is stronger than the one adopted in e-voting protocols (see, e.g., [55]): in particular, the latter assume both the registrar and the tallier (i.e., the credential issuer and the service provider) to be distributed and at least one party for each role to be honest, while CR^{2}P considers a single malicious issuer.

[1] An exception is the attribute-hiding property, which is meaningful only against honest credential issuers, since they know the user's attributes anyway.

[2] In fact, we require that at least one service provider is honest only for coercion-resistance and attribute-hiding. Anonymity and verifiability hold true even if all service providers are malicious.

- a concrete cryptographic instantiation, which we implemented and experimentally evaluated in order to demonstrate the feasibility of our approach.

Outline. The rest of this chapter is organized as follows. We present our definitional framework in Section 4.2. In Section 4.3 we introduce our novel primitive, signatures on randomizable commitments. We describe and explain our main construction of CR^{2}P in Section 4.4. The formal guarantees are summarized in Section 4.5. We describe the implementation and experimental results in Section 4.6. Finally, Section 4.7 discusses the related work.

4.2. Definitional Framework

We start with notational conventions (Section 4.2.1), formally define the concept of a rating system (Section 4.2.2), intuitively describe the security and privacy properties (Section 4.2.3), and discuss the attacker model (Section 4.2.4).

4.2.1. Notation

We denote by $(S_{P_1}, \dots, S_{P_k}) \leftarrow C\langle P_1(in_{P_1}), \dots, P_k(in_{P_k})\rangle$ the protocol C, which is interactively executed by the parties $P_1, \dots, P_k$ where in_x is the input and S_x is the state of $x \in \{P_1, \dots, P_k\}$. The input always implicitly contains a party's state. If S_{P_i} is not affected by the protocol or if in_{P_i} does not exist, we omit it. We use $S\|x$ to denote the concatenation of a state S with x and we assume a set-like structure on states, i.e., an unordered collection of elements. We let τ refer to the protocol-dependent coercion-resistance strategy, which is technically defined as a function taking as input the user's state S and outputting a fake state S'. Intuitively, S and S' should be indistinguishable by the coercer. Moreover, we let $\vec{at}$ denote the list of attributes $(at_1, \dots, at_k)$ and $\vec{at}|_I$ the projection of $\vec{at}$ to the indices in I. We write $\sigma(\vec{at})$ to refer to the credential on $\vec{at}$, which does not reveal any of the attributes $\vec{at}$. We denote by $X_{[n]}$ the sequence $X_1, \dots, X_n$. Finally, we write $x \leftarrow_\$ D$ if x is sampled uniformly at random from D.

4.2.2. Definition of CR^{2}P

We define CR^{2}P as a set of (interactive) protocols executed by a *credential issuer* CI, *users* U, and a *service provider* SP. We follow standard assumptions in the e-voting literature [55, 112] and postulate a distributed SP, i.e., mutually distrustful SP's (e.g., different companies or organizations) that collaborate to offer the rating service. For the remainder of the paper, when we say "the SP's" we refer to the distributed parties behind SP. Furthermore, we assume a trusted bulletin board BB [55], which can be realized in centralized [99] or distributed fashion [62].

The Setup protocol bootstraps the system; the CIReg protocol between CI and U generates an ABC, which is stored by the user; the SPReg protocol between $\mathsf{SP}_1, \dots, \mathsf{SP}_m$ and U generates an identification token, which is stored by the user; the Coerce protocol

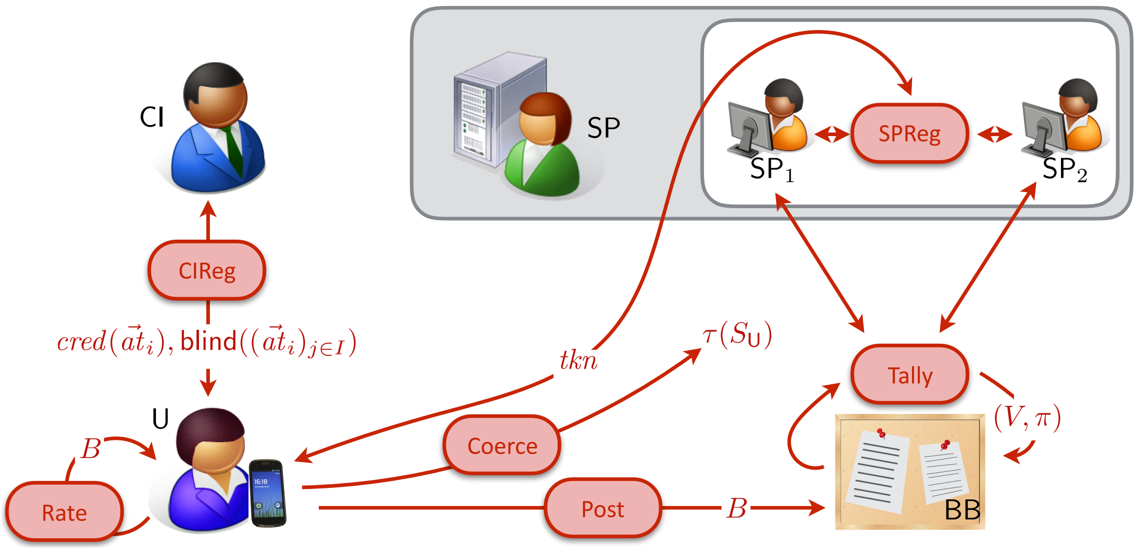


Figure 4.1.: The model for CR^{2}P with entities and functions.

produces a fake state for the coercer; Rate generates a ballot for U formed of the identification token, the credential, and a rating. U uses Post to send the ballot to BB. Finally, the Tally protocol, which is executed by all SP's jointly, publishes valid ratings along with a correctness proof on BB, so that everyone can verify the outcome's correctness. Figure 4.1 illustrates the following definition.

Definition 4.1 (CR^{2}P)**.** *A CR^{2}P consists of the following (interactive)* PPT *protocols, executed between* CI*,* $\mathsf{U}_{[n]}$*, and* $\mathsf{SP}_{[m]}$*:*

$\{S_{\mathsf{CI}}, S_{\mathsf{U}_{[n]}}, S_{\mathsf{SP}_{[m]}}\} \leftarrow \mathsf{Setup}(1^\lambda, svc, I)$: *the setup protocol takes as input the security parameter λ, a service description svc, and a set I of attributes that need to be revealed for eligibility purposes. It outputs the states for all parties.*

$(S_{\mathsf{U}_i} \| \sigma(\vec{at}_i)) \leftarrow \mathsf{CIReg}\langle \mathsf{CI}(\vec{at}_i), \mathsf{U}_i\rangle$: CI *generates the credential $\sigma(\vec{at}_i)$ from the list of attributes $\vec{at}_i$.* U_i *stores the credential.*

$S_{\mathsf{U}_i} \| (tkn_i, tsx_i) \leftarrow \mathsf{SPReg}\langle \mathsf{U}_i, \mathsf{SP}_{[m]}\rangle$: *The* SP*'s and* U_i *establish an identification token tkn_i, which* U_i *stores along with the communication transcript tsx_i. tkn_i is later used to rate.*

$S'_{\mathsf{U}_i} \leftarrow \mathsf{Coerce}(S_{\mathsf{U}_i})$: *the coercion-resistance protocol allows the user to produce a fake state $S'_{\mathsf{U}_i} = \tau(S_{\mathsf{U}_i})$.*

$S_{\mathsf{U}_i} \| B \leftarrow \mathsf{Rate}(\sigma(\vec{at}_i), \vec{at}_i|_I, v, tkn_i)$: *this algorithm generates a ballot B from the credential $\sigma(\vec{at}_i)$, the attributes $\vec{at}_i|_I$, the identification token tkn_i, and a rating v. B is added to* U_i*'s state.*

$\mathsf{BB}' \leftarrow \mathsf{Post}(\mathsf{BB}, B)$: *the ballot posting algorithm appends a ballot B to the bulletin board* BB*, generating an augmented bulletin board* BB'*.*

$\mathsf{BB}\|(V, \pi) \leftarrow \mathsf{Tally}\langle \mathsf{SP}_{[m]}, \mathsf{BB}\rangle$: *the tallying protocol is executed by all* SP*'s on* BB *and publishes the results V of the rating system on* BB *along with a correctness proof π. Usually, V contains the collection of ratings contained in valid ballots.*

In the e-health example, the patient (U) visits the doctor (CI), thereby executing the CIReg protocol. To rate her doctor, the patient contacts the rating service *RateMyDoc* (SP) via SPReg. To compute and submit a rating, the patient uses Rate and Post. Finally, to compute a final, verifiable rating, the distributed computation parties behind *RateMyDoc* run the Tally protocol.

4.2.3. Security and Privacy Properties

4.2.3.1. Adversary Interfaces

In the security and privacy definitions, the adversary has access to a subset of the oracles reported in Figure 4.2. The design of the oracles is driven by the interface offered by CR^{2}P to users, credential issuers, and service providers. Moreover, they use four different sets *Coe*, *Cor*, *Rated*, R_{CI}, and R_{SP} to keep track of the coerced users, the corrupted service providers, the users that already created a ballot, the users registered with CI, and those registered with SP. We treat these variables as global, accessible by any oracle in a game.

4.2.3.2. Attribute-Hiding

Users have to reveal some attributes to let the rating platform decide about eligibility. For example, to rate the doctor, *RateMyDoc* needs to know the date of the medical treatment to judge the currency of a rating. Consequently, all other attributes (which might be necessary for other services or rating platforms) should stay hidden to that platform. The list of required attributes is fixed in the setup protocol for every service provider. Moreover, a coercer who only sees the bulletin board before the Tally protocol is executed, should never learn any attribute, not even if he also corrupted the credential issuer after generation of a credential. Intuitively, we say that a CR^{2}P is attribute-hiding if the adversary cannot extract any attributes from credentials before the Tally protocol is executed and after that point, they only learn those publicly announced in the setup protocol. We allow for user coercion, forcing them to leak their internal state. Incidentally, the service providers could request all attributes of a credential without violating attribute-hiding if the full set of attributes is announced during setup. In order to protect the user, in practice, attribute-hiding has to be combined with accountability. If, for instance, the service provider violates any regulation with his declared attribute set, then this public evidence can be used to hold him accountable for that (e.g., a job employer is not allowed to request the pregnancy status of a female applicant). This is formalized by the following cryptographic game.

```
Setup(I, n, m)
  ${S_{\mathsf{CI}}, S_{\mathsf{U}_{[n]}}, S_{\mathsf{SP}_{[m]}}} \leftarrow \mathsf{Setup}(1^\lambda, I)$
  $R_{\mathsf{CI}}, R_{\mathsf{SP}}, \mathit{Cor}, \mathit{Coe}, \mathit{Rated} \leftarrow \emptyset$
  $\mathsf{BB} \leftarrow [\,], \quad f_{\mathsf{Tally}} \leftarrow 1$
  return ${S_{\mathsf{CI}}, S_{\mathsf{U}_{[n]}}, S_{\mathsf{SP}_{[m]}}, R_{\mathsf{CI}}, R_{\mathsf{SP}}, \mathit{Cor}, \mathit{Coe}, \mathit{Rated}}$

$\mathcal{O}^b_{\mathsf{CIReg}}(i, \vec{at}_i)$
  if $(\forall x.\ (i,x) \notin R_{\mathsf{CI}}) \wedge i \notin \mathit{Coe}$ then
    if $b$ then
      $(S_{\mathsf{U}_i} \| (\sigma(\vec{at}_i)) \leftarrow \mathsf{CIReg}\langle \mathcal{A}(\vec{at}_i), \mathsf{U}_i \rangle$
    else
      $(S_{\mathsf{U}_i} \| (\sigma(\vec{at}_i)) \leftarrow \mathsf{CIReg}\langle \mathsf{CI}(\vec{at}_i), \mathsf{U}_i \rangle$
    $R_{\mathsf{CI}} \leftarrow R_{\mathsf{CI}} \cup \{(i, \vec{at}_i)\}$

$\mathcal{O}_{\mathsf{SPReg}}(i)$
  if $i \notin R_{\mathsf{SP}} \wedge i \notin \mathit{Coe}$ then
    foreach $i \in \mathit{Cor}$ do
      $\mathsf{SP}_i \leftarrow \mathcal{A}$
    $S_{\mathsf{U}_i} \| (\mathit{tkn}_i, \mathit{tsx}_i) \leftarrow \mathsf{SPReg}\langle \mathsf{U}_i, \mathsf{SP}_{[m]} \rangle$
    $R_{\mathsf{SP}} \leftarrow R_{\mathsf{SP}} \cup \{i\}$

$\mathcal{O}_{\mathsf{Rate}}(i, v)$
  if $(\exists x.\ (i,x) \in R_{\mathsf{CI}}) \wedge i \in R_{\mathsf{SP}} \wedge i \notin \mathit{Coe}$ then
    $S_{\mathsf{U}_i} \| B \leftarrow \mathsf{Rate}(\sigma(\vec{at}_i), \vec{at}_i|_I, v, \mathit{tkn}_i)$
    $\mathit{Rated} = \mathit{Rated} \cup \{i\}$

$\mathcal{O}^k_{\mathsf{Corrupt}}(i)$
  if $|\mathit{Cor}| < k$ then
    $\mathit{Cor} \leftarrow \mathit{Cor} \cup \{i\}$
    return $S_{\mathsf{SP}_i}$

$\mathcal{O}_{\mathsf{Post}}(i)$
  if $i \notin \mathit{Coe} \wedge \exists B \in S_{\mathsf{U}_i}$ then
    $B \leftarrow S_{\mathsf{U}_i}$
    $\mathsf{BB} \leftarrow \mathsf{Post}(\mathsf{BB}, B)$

$\mathcal{O}_{\mathsf{Coerce}}(i)$
  if $(\exists x.\ (i,x) \in R_{\mathsf{CI}}) \wedge i \in R_{\mathsf{SP}}$ then
    $\mathit{Coe} \leftarrow \mathit{Coe} \cup \{i\}$
    return $S_{\mathsf{U}_i}$

$\mathcal{O}_{\mathsf{PostBallot}}(X)$
  form $B$ from $X$
  $\mathsf{BB} \leftarrow \mathsf{Post}(\mathsf{BB}, B)$

$\mathcal{O}_{\mathsf{Tally}}(I, I^*)$
  if $f_{\mathsf{Tally}} \wedge I \subseteq I^*$ then
    foreach $i \in \mathit{Cor}$ do
      $\mathsf{SP}_i \leftarrow \mathcal{A}$
    $\mathsf{BB} \| (V, \pi) \leftarrow \mathsf{Tally}\langle \mathsf{SP}_1, \ldots, \mathsf{SP}_m \rangle$
    $f_{\mathsf{Tally}} = 0$
```

Figure 4.2.: The oracles, to a subset of which the adversary has access in the security games.

Definition 4.2 (Attribute-hiding)**.** *An instance of CR^{2}P is* attribute-hiding *if* $\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{AH}}(\lambda, b) = 1\right]$ *is negligibly (in* λ*) close to* $1/2$ *for every* PPT *adversary* $\mathcal{A}$ *where* $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{AH}}(\lambda, b)$ *denotes the experiment in Figure 4.3.*

4.2.3.3. Anonymity

Anonymity means that an attacker has no idea who is the individual behind a rating. Putting it differently, he cannot link ratings to identities. We formalize this intuition as a cryptographic game in which the adversary has to correctly guess who of two challenged users leaves a rating. The adversary may corrupt both the issuer and all but one service provider as well as users of its choice. We report the definition below.

Definition 4.3 (Anonymity)**.** *An instance of CR^{2}P preserves the* anonymity *of its users if* $\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Anon}}(\lambda, b) = 1\right]$ *is negligibly (in* λ*) close to* $1/2$ *for every* PPT *adversary* $\mathcal{A}$ *where* $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Anon}}(\lambda, b)$ *denotes the experiment in Figure 4.4.*

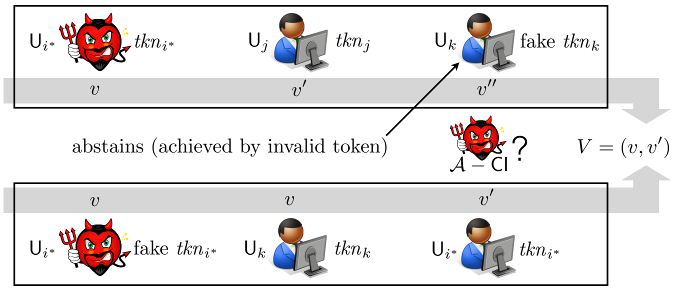
```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{AH}}(\lambda, b)$
$(I, n, m) \leftarrow \mathcal{A}(1^{\lambda})$
$\{S_{\mathsf{CI}}, S_{\mathsf{U}_{[n]}}, S_{\mathsf{SP}_{[m]}}, R_{\mathsf{CI}}, R_{\mathsf{SP}}, Cor, Coe, Rated\} \leftarrow \mathsf{Setup}(I, n, m)$
$/\!/\ \mathbb{O} = \{\mathcal{O}^{0}_{\mathsf{CIReg}}(\cdot,\cdot), \mathcal{O}_{\mathsf{SPReg}}(\cdot), \mathcal{O}^{m-1}_{\mathsf{Corrupt}}(\cdot), \mathcal{O}_{\mathsf{Coerce}}(\cdot), \mathcal{O}_{\mathsf{Rate}}(\cdot,\cdot), \mathcal{O}_{\mathsf{Post}}(\cdot), \mathcal{O}_{\mathsf{PostBallot}}(\cdot)\}$
$(i^*, \vec{at}_0, \vec{at}_1) \leftarrow \mathcal{A}^{\mathbb{O}}()$
$I^* = \{i \mid at_{0,i} = at_{1,i}\}$
if $i^* \notin Coe \wedge i^* \notin R_{\mathsf{CI}}$ then
  $(S_{\mathsf{U}_{i^*}} \| \sigma(\vec{at}_b)) \leftarrow \mathsf{CIReg}\langle \mathsf{CI}(\vec{at}_b), \mathsf{U}_{i^*} \rangle$
  $b' \leftarrow \mathcal{A}^{\mathbb{O} \cup \{\mathcal{O}_{\mathsf{Tally}}(I, I^*)\}}(S_{\mathsf{CI}})$
  if $b = b'$ then
    return 1
return 0
```

Figure 4.3.: The attribute-hiding game $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{AH}}(\lambda, b)$.

4.2.3.4. Coercion-resistance

Intuitively, CR^{2}P is coercion-resistant if the adversary, which in particular controls the credential issuer, cannot force the user to give away her secrets, e.g., to upload a high rating in the rating system on her behalf or to prevent her from rating at all. The idea is that the user can provide the coercer with fake secrets, without the coercer being able to realize that.

More formally, we model coercion resistance as a game between a challenger and an adversary $\mathcal{A}$, following the definition of coercion-resistance for remote electronic voting protocols [16][3]. Intuitively, the adversary has to distinguish the scenario in which he receives the genuine user access token from the one in which he receives a fake token.



If all other users rate the same in both scenarios (see on the right), however, the coercer trivially wins the game, since the faked credential does not allow him to rate and the tallied results look different. Hence, the definition of coercion-resistance includes two additional users, which balance the tallied results. In particular, in the first scenario, the user gives away the real token and the coercer's rating is v, the second user rates v' that the coerced user would wish to rate, and a third user leaves a rating with an invalid token. In the second scenario, the coercer receives a fake token and the coerced

[3]This definition is shown to imply receipt-freeness, resistance to forced-abstention attacks, and vote privacy.

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Anon}}(\lambda, b)$
$(I, n, m) \leftarrow \mathcal{A}(1^\lambda)$
$\{S_{\mathsf{CI}}, S_{\mathsf{U}_{[n]}}, S_{\mathsf{SP}_{[m]}}, R_{\mathsf{CI}}, R_{\mathsf{SP}}, Cor, Coe, Rated\} \leftarrow \mathsf{Setup}(I, n, m)$
// $\mathbb{O} = \{\mathcal{O}^1_{\mathsf{CIReg}}(\cdot,\cdot), \mathcal{O}_{\mathsf{SPReg}}(\cdot), \mathcal{O}^m_{\mathsf{Corrupt}}(\cdot), \mathcal{O}_{\mathsf{Coerce}}(\cdot), \mathcal{O}_{\mathsf{Rate}}(\cdot,\cdot), \mathcal{O}_{\mathsf{Post}}(\cdot), \mathcal{O}_{\mathsf{PostBallot}}(\cdot)\}$
$(i_0, i_1, v) \leftarrow \mathcal{A}^{\mathbb{O}}(S_{\mathsf{CI}})$
if $i_0 \notin Coe \wedge i_1 \notin Coe \wedge i_0 \notin Rated \wedge i_1 \notin Rated$ then
  if $\exists x_0, x_1.\ (i_0, x_0) \in R_{\mathsf{CI}} \wedge (i_1, x_1) \in R_{\mathsf{CI}} \implies x_0|_I = x_1|_I$ then
    if $i_0 \in R_{\mathsf{SP}} \wedge i_1 \in R_{\mathsf{SP}}$ then
      $(S_{\mathsf{U}_{i_b}} \| B) \leftarrow \mathsf{Rate}(\sigma(\vec{at}_{i_b}), \vec{at}_{i_b}|_I, v, tkn_{i_b})$
      $\mathsf{BB} \leftarrow \mathsf{Post}(\mathsf{BB}, B)$
      // $\mathbb{O}'$ defined as $\mathbb{O}$, just that $\mathcal{O}_{\mathsf{Rate}}$ and $\mathcal{O}_{\mathsf{Coerce}}$ are aborted on input $i_0$ or $i_1$
      $b' \leftarrow \mathcal{A}^{\mathbb{O}' \cup \{\mathcal{O}_{\mathsf{Tally}}(I,I)\}}()$
      if $b = b'$ then
        return 1
return 0
```

Figure 4.4.: The anonymity game $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Anon}}(\lambda, b)$.

user rates v', the second user's rating is v, and the third one abstains. In both scenarios, the tallied votes and the number of votes cast with invalid tokens are the same.

One might wonder if it makes sense to assume the existence of a user rating with an invalid access token: this is reasonable in e-voting systems, since there are always users willing to nullify their ballot, but it might look artificial in personal record management systems or, for instance, in rating platforms. Hence, we have to assume that users willingly post ballots with fake credentials on the bulletin board, which look indistinguishable from the ones possibly sent by the coercer based on the faked token (see also the discussion below).

In our definition, the challenger internally runs the users and at least one service provider, while the adversary plays the role of the credential issuer and of the coercer. If $\mathcal{A}$ can correctly determine which scenario the challenger simulates, he wins the game. The definition is reported below.

Definition 4.4 (Coercion-resistance)**.** *An instance of CR^{2}P is* coercion-resistant *if* $\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{CR}}(\lambda, b) = 1\right]$ *is negligibly (in λ) close to* $1/2$ *for every* PPT *adversary $\mathcal{A}$ where* $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{CR}}(\lambda, b)$ *denotes the experiment in Figure 4.5.*

Discussion: rating with fake tokens. As previously mentioned, the coercion-resistance property assumes the existence of a user who rates with a fake token. We support these ratings by real users easily: users can use their real credential but a faked token to submit such a rating. It is then eliminated from the final tally in the fourth step, so it is not

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{CR}}(\lambda, b)$
$(I, n, m) \leftarrow \mathcal{A}(1^\lambda)$
$\{S_{\mathsf{CI}}, S_{\mathsf{U}_{[n]}}, S_{\mathsf{SP}_{[m]}}, R_{\mathsf{CI}}, R_{\mathsf{SP}}, Cor,$
    $Coe, Rated\} \leftarrow \mathsf{Setup}(I, n, m)$
// $\mathbb{O} = \{\mathcal{O}^1_{\mathsf{CIReg}}(\cdot,\cdot), \mathcal{O}_{\mathsf{SPReg}}(\cdot), \mathcal{O}^{m-1}_{\mathsf{Corrupt}}(\cdot),$
//     $\mathcal{O}_{\mathsf{Coerce}}(\cdot), \mathcal{O}_{\mathsf{Rate}}(\cdot,\cdot), \mathcal{O}_{\mathsf{Post}}(\cdot), \mathcal{O}_{\mathsf{PostBallot}}(\cdot)\}$
$i^* \leftarrow \mathcal{A}^{\mathbb{O}}(S_{\mathsf{CI}})$
if $i^* \notin Coe \wedge i^* \notin Rated \wedge i^* \in R_{\mathsf{SP}}$
    $\wedge\, (\exists x.\ (i^*, x) \in R_{\mathsf{CI}})$ then
    $\mathsf{st} \leftarrow S_{\mathsf{U}_{i^*}}$
    if $b$ then
        $\mathsf{st} \leftarrow \tau(S_{\mathsf{U}_{i^*}})$
    // $\mathbb{O}' = \mathbb{O} \cup \{\mathcal{O}()\}$
    $b' \leftarrow \mathcal{A}^{\mathbb{O}'}(\mathsf{st})$
    if $b = b'$ then
        return 1
return 0
```

```
$v \leftarrow \mathsf{Extract}(\mathsf{BB}, i)$
// assume $\mathsf{BB} = (B_1, \ldots, B_k)$ for some $k$
for $k \geq j \geq 1$ do
    if $B_j$ is a ballot for $\mathsf{U}_i$ then
        return $v$, the rating in $B_j$
```

```
$\mathcal{O}()$
$v \leftarrow \mathsf{Extract}(\mathsf{BB}, i^*)$
$v', v'' \leftarrow\!\$\, C$
$j, k \leftarrow\!\$\, \{[n] \mid \{j,k\} \subseteq R_{\mathsf{SP}}$
    $\wedge\ \{j,k\} \cap Rated = \emptyset$
    $\wedge\ \{j,k\} \cap Coe = \emptyset\ \wedge$
    $(\exists x_j, x_k.\ \{(j, x_j), (k, x_k)\} \subseteq R_{\mathsf{CI}}$
        $\implies x_j|_I = x_k|_I = \vec{at}_{i^*}|_I)\}$
if $b\ \wedge\ \exists j,k$ then   // $\mathsf{st} = \tau(S_{\mathsf{U}_{i^*}})$
    $(S_{\mathsf{U}_k} \| B') \leftarrow \mathsf{Rate}(\sigma(\vec{at}_k), \vec{at}_k|_I, v, tkn_k)$
    $(S_{\mathsf{U}_{i^*}} \| B'') \leftarrow \mathsf{Rate}(\sigma(\vec{at}_{i^*}), \vec{at}_{i^*}|_I, v', tkn_{i^*})$
else if $\neg b\ \wedge\ \exists j,k$ then   // $\mathsf{st} = S_{\mathsf{U}_{i^*}}$
    $(S_{\mathsf{U}_j} \| B') \leftarrow \mathsf{Rate}(\sigma(\vec{at}_j), \vec{at}_j|_I, v', tkn_j)$
    // let $tkn'_k$ be a fake token
    $(S_{\mathsf{U}_k} \| B'') \leftarrow \mathsf{Rate}(\sigma(\vec{at}_k), \vec{at}_k|_I, v'', tkn'_k)$
if $\exists j,k$ then
    $\mathsf{BB} \leftarrow \mathsf{Post}(\mathsf{BB}, B')$
    $\mathsf{BB} \leftarrow \mathsf{Post}(\mathsf{BB}, B'')$
    foreach $i \in Cor$ do
        $\mathsf{SP}_i \leftarrow \mathcal{A}$
    $\mathsf{BB} \| (V, \pi) \leftarrow \mathsf{Tally}\langle \mathsf{SP}_1, \ldots, \mathsf{SP}_m \rangle$
```

Figure 4.5.: The game $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{CR}}(\lambda, b)$.

even considered a duplicate vote. Consequently, the assumptions of the coercion-resistance definition can be fulfilled.

One might think that it is too much of a burden for users to rate with fake tokens additionally to their real ratings. Depending on a real deployment, we can envision different solutions. For instance, given that the user trusts the web page or the rating client, that client could inject such fake ratings automatically without the user needing to care of that action. Likewise, a click on an embedded button could allow for the same action, however, explicitly triggered by the respective user.

Discussion: a different way of defining coercion-resistance. We would like to mention another approach of defining coercion-resistance. A key ingredient of all such definitions is the uncertainty of the adversary with respect to some of the ratings. We capture that by requiring a one-to-one correspondence of three different voters such that the overall tallying result is the same no matter how the bit is chosen. Originally, as proposed by

Juels et al. [112], this uncertainty can also be achieved by letting the adversary only trigger the ratings of coerced users. As opposed to this approach, we let the adversary trigger also the ratings of honest users, so that we have to generate the uncertainty in a different way. In any case, the meaning of coercion-resistance is captured by both kinds of definitions.

4.2.3.5. Strong End-to-End Verifiability

Strong end-to-end verifiability [28, 56] is a property that consists of three conditions: (1) individual verifiability with respect to both cast-as-intended and recorded-as-cast verifiability, (2) universal verifiability, and (3) resilience against clash-attacks. We define each of these properties in the following.

Individual Verifiability. Individual verifiability intuitively says that every user can check that, if she intended to leave a certain rating and a certain ballot is posted for her on the bulletin board, this ballot is first really containing the intended rating and second is considered in the tallying process. This is not necessarily a property that needs to be enforced cryptographically, but rather as a safety property. Since this property can be enforced typically by probing the rating client on the generated ballot [1] (cast-as-intended) and by equality checks on the ballots posted on the bulletin board (recorded-as-cast), we stick to the logically formulated definition of [56], arguing later on why our construction trivially achieves recorded-as-cast verifiability and can be extended to also achieve cast-as-intended verifiability.

Universal Verifiability. Universal verifiability [56] guarantees that everyone can check the validity of the final output generated by the Tally protocol. Stated differently, all ratings announced in the final tally belong to valid ballots on the initial bulletin board where duplicates have been removed. We define the predicate $\mathsf{Valid}_{\mathsf{BB}}^{\mathsf{RV}}(B, v, i)$ which holds if B is a *valid* ballot on BB, contains the rating v, belongs to user U_i, and is a non-duplicate with respect to some re-voting policy RV. What *valid* means, has to be defined for a concrete instantiation of CR^{2}P individually.

Definition 4.5. *An instance of CR^{2}P is* universally verifiable *for a re-voting policy* RV *if and only if, for every* PPT *adversary $\mathcal{A}$, the probability that* $\mathsf{Exp}_{\mathsf{VER}}^{\mathcal{A}}(\lambda) = 1$ *is negligible in λ where* $\mathsf{Exp}_{\mathsf{VER}}^{\mathcal{A}}(\lambda)$ *is the experiment in Figure 4.6.*

Note that universal verifiability and coercion-resistance are not contradictory, as the coercer learns that his vote is counted *if and only if* it comes with a valid credential and token, which he does not know upfront.

Resilience against Clash-Attacks. A clash attack is an attack in which two voters are convinced that the same ballot B belongs to each of them which allows a dishonest party to add additional votes of its choice. Much as individual verifiability, we stick to [56] for a formal treatment of clash-attacks and argue why our construction does not suffer this attack later on. The reason is again quite simple: users know what they submit and it is highly unlikely that two users submit the same ballot, hence, equality checks suffice. No two users will believe that one and the same ballot belong to either of them.

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{VER}}(\lambda)$
$(I, n, m) \leftarrow \mathcal{A}(1^\lambda)$
$\{S_{\mathsf{CI}}, S_{\mathsf{U}_{[n]}}, S_{\mathsf{SP}_{[m]}}, R_{\mathsf{CI}}, R_{\mathsf{SP}}, Cor, Coe, Rated\} \leftarrow \mathsf{Setup}(I, n, m)$
// $\mathbb{O} = \{\mathcal{O}^1_{\mathsf{CIReg}}(\cdot,\cdot), \mathcal{O}_{\mathsf{SPReg}}(\cdot), \mathcal{O}^m_{\mathsf{Corrupt}}(\cdot), \mathcal{O}_{\mathsf{Coerce}}(\cdot)\mathcal{O}_{\mathsf{Rate}}(\cdot,\cdot), \mathcal{O}_{\mathsf{Post}}(\cdot), \mathcal{O}_{\mathsf{PostBallot}}(\cdot), \mathcal{O}_{\mathsf{Tally}}(I, I)\}$
$x \leftarrow \mathcal{A}^{\mathbb{O}}(S_{\mathsf{CI}})$
$(B_1, \ldots, B_k, (V, \pi)) \leftarrow \mathsf{BB}$   // $\mathsf{BB}$ after $\mathsf{Tally}$ is run
if $\pi$ verifies on $V$ and $(B_1, \ldots, B_k)$ then
  $V' \leftarrow [\,]$
  for $1 \leq i \leq k$ do
    if $\exists v', j.\ \mathsf{Valid}^{\mathsf{RV}}_{\mathsf{BB}}(B_i, v', j)$ then
      $V' = V' \| (v', j)$
  $V'' = \mathsf{RV}(V')$   // apply $\mathsf{RV}$ to $V'$ and strip off user identifiers
  if $V'' \neq V$ then
    return 1
return 0
```

Figure 4.6.: The game $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{VER}}(\lambda)$.

Entity	Attribute-hiding	Anonymity	Coercion-resistance	Verifiability
CI	C	M	M	M
SP	$m-1$	m	$m-1$	m
U	M	M	M	M

Table 4.1.: The attacker model. We use M for malicious (resp. H for honest and C for compromised). For SP, we indicate how many out of m may be malicious. Finally, if any two parties are malicious, then they may also collude.

4.2.4. Attacker Model

The attacker model is summarized in Table 4.1. As described above, for *attribute-hiding* we assume CI to be honest since it knows the attributes anyway and could simply share them at will with other parties. However, CI may be corrupted after credential issuing; in that case, the attributes that have so far been certified should stay hidden, even if the adversary additionally has CI's secret material. Since some attributes have to be shown for authorization purposes (e.g., the age in an election), we need to assume that at least one service provider is honest since otherwise these attributes could be leaked even before the Tally protocol is executed. Finally, users may be arbitrarily malicious, which captures that they might be corrupted and leak their internal state. *Anonymity* has to hold for honest users even in the presence of all other parties being malicious. *Verifiability* has to hold even if all parties are malicious. For *coercion-resistance* we assume the credential issuer to be malicious but that at least one of the service providers is honest. Malicious parties may coerce users and try to rate on their behalf. Notice that if all service providers were

$$\begin{tikzcd}
 & \vec{m} \arrow[dl, "\mathsf{sign_C}(sk,\cdot,r)"'] \arrow[dr, "\mathsf{sign}(sk,\cdot)"] & \\
(\sigma, oi, aux) \arrow[loop above, "{\mathsf{rand_S}(\cdot, vk, r),\ \mathsf{rand_C}(\cdot, vk, s)}"] \arrow[d, "\mathsf{vfy_{start}}(\cdot, vk, \vec{m}|_R)"'] \arrow[drr, "\mathsf{vfy_C}(\cdot, vk, \vec{m}, \cdot)"] & & \sigma \arrow[loop above, "\mathsf{rand_S}(\cdot, vk, r)"] \arrow[ll, "\mathsf{comm}(sk,\cdot,r)"] \arrow[d, "\mathsf{vfy}(\cdot, vk, \vec{m})"] \\
\hat{\sigma} \arrow[rr, "\mathsf{vfy_{fin}}(\cdot, vk, \vec{m}|_{\bar{R}}, oi)"'] & & \{\top, \bot\}
\end{tikzcd}$$

Figure 4.7.: The functionality diagram of SRC.

malicious, since at some point they have to decide about eligibility and identify ballots originating from coercers, they could simply check whether a coerced user misbehaved with respect to how they told him to act.

This attacker model is strictly stronger than the one typically used in e-voting, where the registrar (i.e., the issuer) is also distributed and at least one has to be honest. As previously argued, this model is unrealistic for more general settings than e-voting since, for instance, it is not clear how to distribute the doctor in the e-health scenario. Moreover, while in e-voting anonymity also assumes that one registrar and tallier are honest, we allow all parties to be malicious.

As in e-voting systems, we assume that user coercion takes place after the user completed both registration protocols: however, when coerced, she has to show the registration transcripts. In general, we believe that without this assumption, it is not possible to get meaningful coercion-resistance guarantees. The intuitive reason is that if the adversary can coerce during registration then he has the power to register himself, which would in turn prevent any kind of countermeasures to detect a coercer. As we will see later, our concrete construction in the setting of multiple CI's and multiple SP's allows the user to give up its identity after coercion, i.e., the identity is publicly revoked so that service providers can check revoked users before issuing tokens. Clearly, this countermeasure is only realistic if coercion is not happening in a strong form such as threatening a person to death in case she is detected. Instead, giving up the identity can make sense in cases of rate buying or forced attribute disclosure by a potential employer. Finally, if there is only one CI and one distributed SP, which is typically the case in e-voting, then even stronger coercion can be tolerated since the honest registration assumption can be easily justified.

4.3. Signatures on Randomizable Commitments

Achieving attribute-hiding and anonymity in the presence of coercion is one of the core challenges of our problem statement. In the ABC realm, users usually prove knowledge of all attributes that are not necessary to be shown to the receiver. Hence, when coerced, the ability to produce a verifying proof of knowledge ultimately breaks attribute-hiding.

We are thus interested in a primitive that allows a user to do something similar but without revealing anything the coercer does not know already, and, at the same time, without necessarily knowing all attributes. A possible solution to this problem could be signatures on randomizable ciphertexts [30], as deployed in BeleniosRF [45]. In our case, the attribute would be encrypted for the SP's and then the user would randomize the ciphertext before sending it to the SP. This has the disadvantage that every signature signs only one message, but we have several attributes that we would like to include in one non-malleable credential. It is unclear how that could be realized with this kind of signatures. Furthermore, the ciphertext targets a specific receiver, which hinders open-ended applications.

Our ideal primitive has the following features: (1) the signature hides the messages, (2) the messages can be opened with a trapdoor that is independent from the identity of the receiver, (3) the trapdoor can be changed, thereby automatically randomizing the signature, and (4) the scheme supports message vectors. To achieve this goal, we introduce the concept of *signatures on randomizable commitments* (SRC), defining their security and privacy properties and presenting a concrete instantiation (see Section 4.6.1).

Definition. We refer to Figure 4.7 for easing the presentation. In the definition, σ denotes a signature, oi the trapdoor (opening information), aux some auxiliary information, and r, s random values. The rest of the notation is standard.

Definition 4.6 (Signature on randomizable commtiments)**.** *A signature scheme on randomizable commitments Π_{SRC} consists of the following* PPT *algorithms:*

$(vk, sk) \leftarrow \mathsf{setup}_{\mathsf{SRC}}(1^\lambda, n)$: *the setup algorithm on input the security parameter λ and an integer n outputs a key pair (vk, sk);*

$\sigma \leftarrow \mathsf{sign}(sk, \vec{m})$: *the signing algorithm on input the signing key sk and a message vector $\vec{m}$ outputs a signature σ;*

$(\sigma, oi, aux) \leftarrow \mathsf{sign}_{\mathsf{C}}(sk, \vec{m}, r)$: *the commitment-signing algorithm takes as input the signing key sk, a message vector $\vec{m}$, and a random value r and outputs a signature σ, the opening information for the commitment oi, and auxiliary information aux;*

$(\sigma', oi, aux) \leftarrow \mathsf{comm}(sk, \sigma, r)$: *the commitment algorithm takes as input the signing key sk, a signature σ, and a random value r and it outputs an adapted signature on a commitment σ', opening information for the commitment oi, and auxiliary information aux;*

$\{\top, \bot\} \leftarrow \mathsf{vfy}(\sigma, vk, \vec{m})$: *the verification algorithm on input a signature σ, the verification key vk, and a message vector $\vec{m}$ outputs either $\top$ in case of success or $\bot$ in case of failure;*

$\{\top, \bot\} \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma, vk, \vec{m}, oi)$: *the commitment-verification algorithm takes as input a signature σ, the verification key vk, a message vector $\vec{m}$, and opening information oi and it outputs either $\top$ in case of success or $\bot$ in case of failure;*

$\hat{\sigma} \leftarrow \mathsf{vfy}_{\mathsf{start}}(\sigma, vk, \vec{m}|_R)$: *the verification-starting algorithm takes as input a signature σ, the verification key vk, and a partial message vector $\vec{m}|_R$ and it outputs a reduced signature $\hat{\sigma}$, the verification of which is started;*

$\{\top, \bot\} \leftarrow \mathsf{vfy}_{\mathsf{fin}}(\hat{\sigma}, vk, \vec{m}|_{\bar{R}}, oi)$: *the verification-finish algorithm takes as input a reduced signature $\hat{\sigma}$, the verification key vk, a partial message vector $\vec{m}|_{\bar{R}}$, and opening information oi and it outputs either $\top$ in case of success or $\bot$ in case of failure;*

$(\sigma', [aux']) \leftarrow \mathsf{rand}_{\mathsf{S}}(\sigma, [aux], vk, r)$: *the signature-randomization algorithm takes as input a signature σ, optionally auxiliary information aux, the verification key vk, and a random value r and it outputs a randomized signature σ' and optionally adapted auxiliary information aux′ (in case aux was given originally as input);*

$(\sigma', oi') \leftarrow \mathsf{rand}_{\mathsf{C}}(\sigma, aux, vk, s)$: *the commitment-randomization algorithm takes as input a signature σ, auxiliary information aux, the verification key vk, and a random value s and it outputs a randomized signature σ' and opening information oi′, which can be used to adapt the opening information of the commitment inside the signature.*

Assume that $\mathsf{Valid}(\sigma, aux)$ holds whenever σ and aux belong together, i.e., they have been output together either by comm, $\mathsf{sign}_{\mathsf{C}}$, or $\mathsf{rand}_{\mathsf{S}}$. Furthermore, if $\mathsf{Valid}(\sigma, aux)$ holds then for any call to $\mathsf{rand}_{\mathsf{C}}$, which outputs a new σ', we have $\mathsf{Valid}(\sigma', aux)$. With this assumption in mind, a signature scheme on randomizable commitments is *complete* if for all λ, n, $(vk, sk) \leftarrow \mathsf{setup}_{\mathsf{SRC}}(1^\lambda, n)$, and all $\vec{m}$, r, s, oi, oi', aux, aux', σ, σ', $\hat{\sigma}$, and R the following holds:

$$\sigma \leftarrow \mathsf{sign}(sk, \vec{m}) \implies \top \leftarrow \mathsf{vfy}(\sigma, vk, \vec{m}) \tag{4.8}$$

$$\top \leftarrow \mathsf{vfy}(\sigma, vk, \vec{m}) \ \wedge \ \sigma' \leftarrow \mathsf{rand}_{\mathsf{S}}(\sigma, vk, r) \implies \top \leftarrow \mathsf{vfy}(\sigma, vk, \vec{m}) \tag{4.9}$$

$$\top \leftarrow \mathsf{vfy}(\sigma, vk, \vec{m}) \ \wedge \ (\sigma', aux, oi) \leftarrow \mathsf{comm}(sk, \sigma, r) \implies \top \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma', vk, \vec{m}, oi) \tag{4.10}$$

$$(\sigma, oi, aux) \leftarrow \mathsf{sign}_{\mathsf{C}}(sk, \vec{m}, r) \implies \top \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma, vk, \vec{m}, oi) \tag{4.11}$$

$$\top \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma, vk, \vec{m}, oi) \ \wedge \ \mathsf{Valid}(\sigma, aux) \ \wedge \ (\sigma', aux') \leftarrow \mathsf{rand}_{\mathsf{S}}(\sigma, aux, vk, r) \implies \top \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma', vk, \vec{m}, oi) \tag{4.12}$$

$$\top \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma, vk, \vec{m}, oi) \ \wedge \ \mathsf{Valid}(\sigma, aux) \ \wedge \ (\sigma', oi') \leftarrow \mathsf{rand}_{\mathsf{C}}(\sigma, aux, vk, s) \implies \top \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma', vk, \vec{m}, oi + oi') \tag{4.13}$$

$$\top \leftarrow \mathsf{vfy}_{\mathsf{C}}(\sigma, vk, \vec{m}, oi) \ \wedge \ \hat{\sigma} \leftarrow \mathsf{vfy}_{\mathsf{start}}(\sigma, vk, \vec{m}|_R) \implies \top \leftarrow \mathsf{vfy}_{\mathsf{fin}}(\hat{\sigma}, vk, \vec{m}|_{\bar{R}}, oi) \tag{4.14}$$

Key Idea of Using SRC. SRCs allow users to prove their eligibility to rate according to the authorization policy in place on the rating platform. Intuitively, the ABC is formed of a collection of SRCs, each signing the user's blinded identity bid_{U} and an attribute at_i, committed to using randomness s_i. The ABC is completed by the auxiliary information aux_i as well as the opening information oi_i in encrypted form for SP.

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{EUF-CMA}}(\lambda, n)$
$(vk, sk) \leftarrow \mathsf{setup}_{\mathsf{SRC}}(1^\lambda, n)$
$(\sigma^*, [oi^*], \vec{m}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(vk)$
// $[oi^*]$ means $oi^*$ is optional
if $\top = \mathsf{vfy}(\sigma^*, vk, \vec{m}^*)$
    $\vee \top = \mathsf{vfy}_{\mathsf{C}}(\sigma^*, vk, \vec{m}^*, oi^*)$
    if $\vec{m}^*$ not queried to $\mathcal{O}$
        return 1
return 0
```

```
$\mathsf{Exp}^{\mathcal{A}}_{\mathsf{HID}}(\lambda, b, n)$
$(vk, sk) \leftarrow \mathsf{setup}_{\mathsf{SRC}}(1^\lambda, n)$
$(\vec{m}_0, \vec{m}_1, S) \leftarrow \mathcal{A}^{\mathcal{O}}(vk)$
choose $r$ uniformly at random
$(\sigma^*, oi^*, aux^*) \leftarrow \mathsf{sign}_{\mathsf{C}}(sk, \vec{m}_b, r)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(S, \sigma^*, aux^*, sk)$
if $|\vec{m}_0| = |\vec{m}_1| = n \ \wedge \ b = b'$
    return 1
return 0
```

Figure 4.15.: Games for unforgeability (left) and message-hiding (right) where $\mathcal{O} = (\mathsf{sign}(sk, \cdot), \mathsf{sign}_{\mathsf{C}}(sk, \cdot, \cdot), \mathsf{comm}(sk, \cdot, \cdot))$.

In order to rate, the user randomizes the SRCs corresponding to the required attributes, changes the opening information, and starts their verification against her identity by means of $\mathsf{vfy}_{\mathsf{start}}$. It sends then the half-verified SRCs along with the encrypted, changed, and randomized opening information and the encrypted attributes to BB. The SP's can then finish the verification of the SRCs ($\mathsf{vfy}_{\mathsf{fin}}$) using the decrypted attributes and opening information.

Security. The security notion of SRC is equivalent to existential unforgeability against chosen message attacks (EUF-CMA) with the difference that in addition to a sign oracle, the adversary gets also access to a commit and sign-commit oracle. His goal is to produce a signature or signed commitment on a message that has not been queried to any of these oracles. This notion unifies signature unforgeability and the binding property of commitment schemes.

Definition 4.7 (EUF-CMA)**.** *A SRC is* existentially unforgeable against adaptive chosen message attacks (EUF-CMA) *if any* PPT *adversary $\mathcal{A}$ has a negligible probability of winning the experiment $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{EUF-CMA}}(\lambda, n)$ in Figure 4.15 (left).*

Privacy. Privacy of SRC is similar to commitment hiding. In the game below, the adversary has to decide which of two message vectors is signed in committed form. Of course, he does not learn the opening information, preventing him from opening the commitment.

Definition 4.8 (Message-hiding)**.** *A SRC is* message-hiding *if any* PPT *adversary $\mathcal{A}$ has a probability negligibly close to $1/2$ of winning the experiment $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{HID}}(\lambda, b, n)$ in Figure 4.15 (right).*

4.4. Our Construction

In this section we introduce the cryptographic building blocks (Section 4.4.1) and system assumptions (Section 4.4.2), and then we present our cryptographic realization of CR^{2}P

Scheme	**Notation**
SRC	Π_{SRC}, see Section 4.3
Public-key encryption	Π_{PKE}, $\Pi_{\mathsf{PKE}}{}^{+}$
Key-generation	$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^{\lambda})$
Enc./Dec.	$c \leftarrow \mathsf{E}(ek, m)$, $m \leftarrow \mathsf{D}(dk, c)$
Randomization	$c' \leftarrow \mathsf{Rnd}(ek, c, r)$
Distributed key-gen	$ek \leftarrow \mathsf{GenEK}(ek_1, \ldots, ek_{\ell})$
Partial decryption	$D_i \leftarrow \mathsf{PartD}(dk_i, c)$
Distributed decryption	$m \leftarrow \mathsf{DistD}(D_1, \ldots, D_{\ell}, c)$
PET	$(\{\top, \bot\}, \pi) \leftarrow \mathsf{PET}(dk_1, \ldots, dk_{\ell}, c_1, c_2)$
Mult. homomorphic	$c \leftarrow \mathsf{E}(ek, m)$, $d \leftarrow \mathsf{E}(ek, n)$,
	$mn \leftarrow \mathsf{D}(dk, c \otimes d)$,$m^n \leftarrow \mathsf{D}(dk, c^n)$
Add. homomorphic	$c \leftarrow \mathsf{E}^{+}(ek, m)$, $d \leftarrow \mathsf{E}^{+}(ek, n)$,
	$m + n \leftarrow \mathsf{D}^{+}(dk, c \oplus d)$,
	$mn \leftarrow \mathsf{D}^{+}(dk, c \cdot n)$
User identity	$id \leftarrow \mathsf{gen}_{\mathsf{ID}}(1^{\lambda})$
Blinding	$bid \leftarrow \mathsf{blind}(id)$
Blind signing	$\top \leftarrow \mathsf{vfy}(\mathsf{sign}(sk, bid), vk, id)$
SSP	$psd \leftarrow \mathsf{SSP}_{\mathbb{G}}(id, svc)$
NIZK	$P = PK\,\{(\vec{x}):\ F(\vec{x}, \vec{y})\}$
DVP for user ID *id*	$P = DVP_{bid}\,\{(\vec{x}) : F(\vec{x}, \vec{y})\}$
Faked DVP for *id*	$P = DVP^{id}_{bid}\,\{F(\vec{x}, \vec{y})\}$
	$\vec{x}$ hidden by P, $\vec{y}$ revealed by P
Mixing	$(\vec{c}_2, \pi) \leftarrow \mathsf{Mix}(\vec{c}_1)$

Table 4.2.: Notation for cryptographic primitives.

(Section 4.4.3). We end by discussing some challenges when generalizing our scheme (Section 4.4.4).

4.4.1. Cryptographic Primitives

We deploy several standard cryptographic primitives such as multiplicatively (resp. additively) homomorphic public-key encryption with distributed key-generation, distributed decryption and plaintext-equivalence-test (PET) [70, 76, 109, 153, 155], service-specific pseudonyms (SSP) [143], non-interactive zero-knowledge proofs of knowledge (NIZK), designated verifier proofs (DVP) [111], and verifiable mix networks [110, 117, 122, 127] (or proofs of shuffle correctness [22]), as well as SRC. We summarize the notation for all these primitives in Table 4.2 and present required definitions and instantiations in Section C.2.

```
Setup($1^\lambda, I$)

// User setup
foreach $\mathsf{U}_i$ do
    $id_i \leftarrow \mathsf{gen}_{\mathsf{ID}}(1^\lambda)$
    $bid_i \leftarrow \mathsf{blind}(id_i)$
    send $bid_i$ to PKI
    receive $\sigma_{id_i}$ s.t.
        $\top \leftarrow \mathsf{vfy}(\sigma_{id_i}, vk_{\mathsf{PKI}}, id_i)$
    return $S_{\mathsf{U}_i} = (id_i, \sigma_{id_i})$
// CI setup
$(sk_{\mathsf{CI}}, vk_{\mathsf{CI}}) \leftarrow \mathsf{setup}_{\mathsf{SRC}}(1^\lambda, 4)$
return $S_{\mathsf{CI}} = (sk_{\mathsf{CI}}, vk_{\mathsf{CI}})$

// SP setup
for $1 \leq i \leq m$ do
    $(ek_i, dk_i) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$
    $b_i \leftarrow\$ \mathbb{Z}_q^*$
    $B_i = g^{b_i}$
    return $S_{\mathsf{SP}_i} = (ek_i, dk_i, b_i)$
$ek_{\mathsf{SP}} \leftarrow \mathsf{GenEK}(ek_1, \ldots, ek_m)$
$\mathsf{BB} \leftarrow \mathsf{BB}||(ek_{\mathsf{SP}}, I, B_{[m]}, ek_{[m]})$
```

Figure 4.16.: The Setup protocol.

4.4.2. System Assumptions

We assume the existence of a root of trust, concretely a public-key infrastructure PKI. We also assume that all participants are aware of the parameters $(q, \mathbb{G}, g)$ for a finite cyclic group generated by $\mathsf{GSetup}(1^\lambda)$, where $\mathbb{G}$ is a multiplicative group of order q, generated by g. Finally, we assume the existence of an append-only bulletin board BB and a set of fixed rating choices C (ratings in prose cannot be made coercion-resistant).

4.4.3. Realization of CR^{2}P

We present our cryptographic construction of CR^{2}P.

Setup. The setup procedures are reported in Figure 4.16. Users generate identities and register them blindly with the PKI. CI sets up a signature on randomizable commitments with at most four messages. The SP's generate a joint encryption key with distributed decryption keys and a secret value that is used to generate identification tokens and which is posted in public form (B_i) on BB.

Registration with CI. CI generates a credential for user U_i. The credential is a collection of SRCs on the user's attributes. Additionally, the user receives all opening information in encrypted form for SP, restricted to those announced in the set I. Clearly, if there is more than one SP overall (neglecting distributed computation parties behind SP), then the user has to retrieve several different sets of encrypted opening information, one for each SP and restricted to that SP's set I.

Concretely, U_i first sends her blinded identity $bid_i \leftarrow \mathsf{blind}(id_i)$ to CI along with a proof $P_{id} \leftarrow PK\{(\alpha) : \ bid_i = \mathsf{blind}(\alpha) \land \top = \mathsf{vfy}(\sigma_{id_i}, vk_{\mathsf{PKI}}, \alpha)\}$. CI then executes the algorithm in Figure 4.17, where r is a random value, which is part of each sub-credential so

```
$\mathsf{CIReg}\langle \mathsf{CI}(\vec{at_i}), \mathsf{U}_i \rangle : \mathsf{CI}(sk_{\mathsf{CI}}, vk_{\mathsf{CI}})$'s part
1 : verify $P_{id}$ for $bid_i$
2 : $r, r_1, \ldots, r_\ell \leftarrow\!\!\$\, \mathbb{Z}_q^*$
3 : $\sigma_i^0 \leftarrow \mathsf{sign}(sk_{\mathsf{CI}}, (bid_i, r))$
4 : for $1 \leq j \leq \ell$ do
5 :     $(\sigma_i^j, aux_i^j, oi_i^j) \leftarrow \mathsf{sign_C}(sk_{\mathsf{CI}}, (bid_i, r, at_i^j), r_j)$
6 : $\bar{oi}_i^{\mathsf{SP}} = \{\mathsf{E}^+(ek_{\mathsf{SP}}, oi_i^j)\}_{j \in I}$
7 : $\pi \leftarrow DVP_{bid_i} \left\{ \begin{array}{l} (\alpha, \beta_1, \ldots, \beta_\ell, \gamma_1, \ldots, \gamma_\ell) : \\ \sigma_i^0 \leftarrow \mathsf{sign}(\alpha, (bid_i, r)) \wedge \\ \forall 1 \leq j \leq \ell.\ (\sigma_i^j, aux_i^j, \beta_j) \leftarrow \mathsf{sign_C}(\alpha, (bid_i, r, at_i^j), \gamma_j) \wedge \\ \quad (j \in I \implies (\bar{oi}_i^{\mathsf{SP}})_j = \mathsf{E}^+(ek_{\mathsf{SP}}, \beta_j)) \end{array} \right\}$
8 : $\sigma = (\{\sigma_i^j\}_{0 \leq j \leq \ell}, \{aux_i^j\}_{1 \leq j \leq \ell}, r, \bar{oi}_i^{\mathsf{SP}}, \pi)$
```

Figure 4.17.: The CIReg protocol.

as to prevent mixing attacks.[4] Furthermore, the proof π is necessary to enforce an honest behavior on CI, otherwise it could send malformed credentials without the user noticing. Notice also that π reveals the attributes, which is okay, since π is a designated verifier proof and the user can fake it, replacing the shown attributes with different (and maybe wrong) ones of her choice. This is sufficient to guarantee attribute-hiding. CI then sends σ to U_i, who extracts and verifies π on the other components of σ, and finally appends $\{\sigma_i^j\}_{0 \leq j \leq \ell}$, $\{aux_i^j\}_{1 \leq j \leq \ell}$, r, and $\bar{oi}_i^{\mathsf{SP}}$ to S_{U_i}.

Registration with SP. In this protocol, the user U_i obtains an identification token which is used later on in the rating procedure at SP for service *svc* (describing the rating target, e.g., the specific doctor). In a nutshell, such a token is the encryption of a user's SSP for *svc*, blinded by all SP_j's random values b_j. This real token allows the user to leave a rating without revealing her identity but showing that she is eligible. The token is a key ingredient to achieve coercion-resistance as a coercer receives only a faked version thereof, which he cannot distinguish from the real one. Figure 4.18 details the protocol, which is executed by each SP_j individually.

Coercion. In this algorithm, the user produces a state which can be handed out to the coercer, however, thereby allowing the coercer neither to rate nor to recognize this inability. To this end, remember that at least one service provider has to be honest and furthermore, this algorithm is executed only after the above registration protocols have been run. Concretely, the algorithm (Figure 4.19) includes the real information for the user except the identification token and the proof of correct credential generation, which are both faked.

[4]This feature is only relevant if there are more than one registration allowed for a single user. Without this protection, the user could compose a faked "credential" formed of attributes of her choice, extracted from different valid "credentials".

```
$\mathsf{SPReg}\langle \mathsf{U}_i, \mathsf{SP}_{[m]} \rangle$
**for** $1 \leq j \leq m$ **do**
    $s_j \leftarrow_{\$} \mathbb{Z}_q^*$
    $bid_i \leftarrow \mathsf{blind}(id_i)$
    $tkn_j \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(id_i, svc))^{s_j}$
    $P_j^1 \leftarrow PK\left\{(\alpha) : bid_i \leftarrow \mathsf{blind}(\alpha) \wedge \top = \mathsf{vfy}(\sigma_{id_i}, vk_{\mathsf{PKI}}, \alpha)\right\}$
    $\xrightarrow{(bid_i, tkn_j, P_j^1)}$
        $\mathsf{SP}_j$
        check $P_j^1$
        $tkn' = tkn_j^{b_j}$
        $P_j^2 = DVP_{bid_i}\left\{(\gamma) : B_j = g^{\gamma} \wedge tkn' = tkn^{\gamma}\right\}$
        **return** $(tkn', P_j^2)$
    check $P_j^2$
    $tkn_i^j = tkn'^{s_j^{-1}}$
    $S_{\mathsf{U}_i} \leftarrow S_{\mathsf{U}_i} || (tkn_i^j, tkn_j, s_j, P_j^2)$
**endfor**
$tkn_i = \bigotimes_{j=1}^{m} tkn_i^j$
$S_{\mathsf{U}_i} \leftarrow S_{\mathsf{U}_i} || tkn_i$
```

Figure 4.18.: The SPReg protocol.

```
Coerce(i)
1 :  $S'_{\mathsf{U}_i} = id_i || \{\sigma_i^j\}_{0 \le j \le \ell} || \{aux_i^j\}_{1 \le j \le \ell} || r || \bar{oi}_i^{\mathsf{SP}}$
2 :  $\{at_F^j\}_{1 \le j \le \ell}$    // fake attributes
3 :  $\pi^F \leftarrow DVP_{bid_i}^{id_i} \left\{ \begin{array}{l} (\alpha, \beta_1, \ldots, \beta_\ell, \gamma_1, \ldots, \gamma_\ell) : \\ \quad \sigma_i^0 \leftarrow \mathsf{sign}(\alpha, (bid_i, r)) \wedge \\ \quad \forall 1 \le j \le \ell.\ (\sigma_i^j, aux_i^j, \beta_j) \leftarrow \mathsf{sign}_{\mathsf{C}}(\alpha, (bid_i, r, at_F^j), \gamma_j) \wedge \\ \quad\quad (j \in I \implies (\bar{oi}_i^{\mathsf{SP}})_j = \mathsf{E}^+(ek_{\mathsf{SP}}, \beta_j)) \end{array} \right\}$
4 :  $S'_{\mathsf{U}_i} = S'_{\mathsf{U}_i} || \pi^F$
5 :  // let $j$ be the index of a trusted SP
6 :  $(tkn_i^j, tkn_j, s_j) \leftarrow S_{\mathsf{U}_i}$
7 :  $c \leftarrow\!\$\ \mathbb{Z}_q^*$    // the fake blinding factor (aka. $b_j$)
8 :  $tkn^F = tkn_j^{s_j c}$
9 :  $P_F^2 = DVP_{bid_i}^{id_i} \left\{ tkn^F = tkn_j^{s_j \gamma} \wedge B_j = g^\gamma \right\}$
10 : $tkn_i^F = tkn_i \otimes (tkn_i^j)^{-1} \otimes tkn^F$
11 : $S'_{\mathsf{U}_i} = S'_{\mathsf{U}_i} || (tkn_i^1, tkn_1, s_1, P_1^2) || \cdots || (tkn^F, tkn_j, s_j, P_F^2) || \cdots || (tkn_i^m, tkn_m, s_m, P_m^2)$
12 : $S'_{\mathsf{U}_i} = S'_{\mathsf{U}_i} || tkn^F$
```

Figure 4.19.: The Coerce algorithm.

Rating. In this protocol (Figure 4.20), the user creates a ballot B which allows her to rate the subject *svc*. B is formed of (1) versions of the credentials from CI which are randomized inside (the commitment) so as to achieve anonymity even against a collusion of CI and all SP_j; (2) the identification token, which prevents coercion and duplicate ratings, (3) an encrypted pseudonym (which can be seen as a freshly started token), which is used to check the validity of the token, (4) the encryption of the actual rating, and finally (5) a proof of B's correct generation.

We employ a general technique proposed by Adida [1] to get cast-as-intended verifiability. Whenever a ballot is generated, the user can either use it or it can probe the software so as to reveal all random values used for generating ciphertexts and zero-knowledge proofs so as to check whether the ballot has been computed from the information given by the user. This step can be repeated arbitrarily often.

Post. To post a ballot B on the public bulletin board BB, the user sends it via an anonymous channel to BB.

Tally. This protocol computes the tally in a way that makes it publicly verifiable. The idea is to take the ballots from BB and to filter out all invalid ones, finally ending up with votes corresponding to valid ballots. Before describing the process step-by-step, we shall define what valid means. To this end, we define the predicate $\mathsf{Valid}_{\mathsf{BB}}^{\mathsf{RV}}(B, v, j)$ below.

Definition 4.9 (Valid ballot)**.** *Let the i-th ballot on* BB *be* $B_i = (\{\hat{\sigma}_i^j, \hat{oi}_i^j, \hat{at}_i^j\}_{j \in I}, \hat{p}_i, \hat{v}_i,$

```
$\mathsf{Rate}(\sigma(\vec{at}_i), \vec{at}_i|_I, v, tkn_i)$
foreach $j \in I$ do
    $s_j \leftarrow_{\$} \mathbb{Z}_q^*$
    $(\tilde{\sigma}_i^j, \tilde{oi}_i^j) \leftarrow \mathsf{rand}_{\mathsf{C}}(\sigma_i^j, aux_i^j, vk_{\mathsf{CI}}, s_j)$
    // assume $\tilde{oi}_i^j$ can be added homomorphically
    $\hat{oi}_i^j = \bar{oi}_i^j \oplus \mathsf{E}^+(ek_{\mathsf{SP}}, \tilde{oi}_i^j)$
    $\hat{at}_i^j \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, at_i^j)$
    $\hat{\sigma}_i^j \leftarrow \mathsf{vfy}_{\mathsf{start}}(\tilde{cert}_i^j, vk_{\mathsf{CI}}, \{id_i, r\})$
$\hat{p}_i \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(id_i, svc))$
$\hat{tkn}_i \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, tkn_i)$
$\hat{v}_i \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, v)$
$P_i \leftarrow PK \left\{ \begin{array}{l} (\sigma_i^0, \{\tilde{\sigma}_i^j\}_{j \in I}, id, r, tkn_i, v, \sigma_{id}) : \\ \quad \top = \mathsf{vfy}(\sigma_{id}, vk_{\mathsf{PKI}}, id) \ \wedge \ \top = \mathsf{vfy}(\sigma_i^0, vk_{\mathsf{CI}}, (id, r)) \ \wedge \\ \quad \forall j \in I.\ \hat{\sigma}_i^j = \mathsf{vfy}_{\mathsf{start}}(\tilde{\sigma}_i^j, vk_{\mathsf{CI}}, \{id, r\}) \\ \quad \wedge\ \hat{p}_i \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(id, svc)) \ \wedge \ \hat{tkn}_i = \mathsf{E}(ek_{\mathsf{SP}}, tkn_i) \ \wedge \\ \quad \hat{v}_i \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, v) \ \wedge \ v \in C \end{array} \right\}$
$B = (\{\hat{\sigma}_i^j, \hat{oi}_i^j, \hat{at}_i^j\}_{j \in I}, \hat{p}_i, \hat{v}_i, \hat{tkn}_i, P_i)$
$S_{\mathsf{U}_i} = S_{\mathsf{U}_i} || B$
```

Figure 4.20.: The Rate protocol.

$\hat{tkn}_i, P_i)$. *Then* $\mathsf{Valid}_{\mathsf{BB}}^{\mathsf{RV}}(B_i, v, j)$ *states that, according to a re-voting policy* RV *and with respect to* BB, B_i *is valid for user* U_j *and rating* v. *Formally,*

1. P_i *verifies with respect to* B_i,
2. *the finishing signature verification succeeds on* $\hat{\sigma}_i^j$ *and the plaintexts of* $\hat{oi}_i^j$ *and* $\hat{at}_i^j$, *respectively,*
3. v *is the rating encrypted in* $\hat{v}_i$,
4. *and finally,* $\hat{tkn}_i$ *is the encryption of* U_j*'s token.*

The validity of a ballot is a local property, i.e., two duplicate ballots can both be valid in the first place. Only after applying RV, one of them is deemed valid, e.g., the last vote submitted considering the chronological order.

To generate the final tally, i.e., the list of votes which belong to valid ballots in the global sense (without duplicates), $\mathsf{SP}_1, \ldots, \mathsf{SP}_m$ jointly carry out the protocol in Figure 4.21. We introduce a shortcut for provable distributed decryption as follows: let c be a ciphertext that is encrypted with ek_{SP}. To provably decrypt, the SP_j jointly carry out the following protocol.

```
$\mathsf{Tally}(\mathsf{SP}_1, \dots, \mathsf{SP}_m)$

1. Ballot verification
for $1 \leq i \leq N_1$ do
    foreach $\mathsf{SP}_j$ do
        verify $P_i$ w.r.t. $B_i$
    $(oi_i^\ell, \pi_{1i}^\ell) \leftarrow \mathsf{DProto}(\mathsf{SP}_{[m]}, \hat{oi}_i^\ell)$
    $(at_i^\ell, \pi_{2i}^\ell) \leftarrow \mathsf{DProto}(\mathsf{SP}_{[m]}, \hat{at}_i^\ell)$
    foreach $\mathsf{SP}_j$ do
        $x_i^\ell = \mathsf{vfy}_{\mathsf{fin}}(\hat{\sigma}_i^\ell, vk_{\mathsf{CI}}, at_i^\ell, oi_i^\ell)$
    if $at_i^\ell$ not eligible $\vee\ x_i^\ell = \perp$ then
        remove $B_i$ from $\mathsf{BB}$
    $(tkn_i, \pi_{3i}) \leftarrow \mathsf{DProto}(\mathsf{SP}_{[m]}, \hat{tkn}_i)$
reduce all ballots to $B_i = (\hat{p}_i, \hat{v}_i, tkn_i)$

2. Duplicate elimination
for $1 \leq i \leq N_2$ do
    $(y_i, \pi_{4i}) \leftarrow \mathsf{DProto}(\mathsf{SP}_{[m]}, tkn_i)$
Apply $\mathsf{RV}$ on the list $(y_1, \dots, y_{N_2})$

3. Mixing
$\vec{B}_0 = (B_1, \dots, B_{N_3})$ // $B_i = (\hat{p}_i, \hat{v}_i, tkn_i)$
foreach $\mathsf{SP}_j$ do
    $(\vec{B}_j, \pi_{5j}) \leftarrow \mathsf{Mix}(\vec{B}_{j-1})$

4. Invalid elimination
for $1 \leq i \leq N_3$ do
    foreach $\mathsf{SP}_j$ do
        $q_i^j = \hat{p}_i^{b_j}$
        $\pi_{6i}^j = PK\left\{(\alpha) :\ q_i^j = \hat{p}_i^\alpha \wedge B_j = g^\alpha\right\}$
    $(z_i, \pi_{7i}) \leftarrow \mathsf{PET}(dk_{[m]}, \bigotimes_{j=1}^{m} q_i^j, tkn_i)$

5. Decrypt ratings
for $1 \leq i \leq N_4$ do
    $(v_i, \pi_{8i}) \leftarrow \mathsf{DProto}(\mathsf{SP}_{[m]}, \hat{v}_i)$
```

Figure 4.21.: The Tally protocol where N_1 is the number of ballots on BB before Tally starts, N_2 are remaining that are accompanied with valid proofs, N_3 are non-duplicates, and N_4 have valid tokens.

```
$(p, \pi) \leftarrow \mathsf{DProto}(\mathsf{SP}_{[m]}, c)$

foreach $\mathsf{SP}_j$ do
    $D_j = \mathsf{PartD}(dk_j, c)$
    $\pi_j = PK\left\{(\alpha) :\ (ek_j, \alpha) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda) \wedge D_j = \mathsf{PartD}(\alpha, c)\right\}$
$p = \mathsf{DistD}(D_1, \dots, D_m, c)$
return $(p, (\pi_1, \dots, \pi_m))$
```

1. Ballot verification: The SP's check conditions one and two of Definition 4.9, i.e., (1) the validity of every B_i's proof P_i and (2) the validity of partially verified signatures with respect to the encrypted attributes and opening information. The SP's then check the validity of partially verified credentials as well as the eligibility of the corresponding attributes. Ballots that do not fulfill any of those requirements are discarded.

With this step we achieve *recorded-as-cast individual verifiability* [28, 56], since the user has to check whether her ballot is present on the initial bulletin board and whether it is considered in the next step, and we achieve *resilience against clash attacks* [126], since an attacker cannot use components of B_i to cast a fresh, valid, and (mistakenly) duplicate ballot since that would require the knowledge of a valid token that must be encrypted by the ballot sender. We also rule out the classical *Italian attack* [57], in which a coercer forces the user to submit a specific invalid vote, since the proof P_i shows the compliance of the encrypted vote with the available vote choices in C.

2. Removing duplicates: We remove duplicates while preserving anonymity. We only reveal which ballots on the board belong to the same user. That allows us to apply RV.

Our technique to achieve $O(N_2)$ complexity is similar to the solution by Weber et al. [191], who blindly decrypt the credential (cf. the identification token in our work). Instead, in CR^{2}P, the SP's just jointly decrypt the token, revealing the user's SSP, which is not known to anyone and is thus uncritical for anonymity on the one hand. The linkability of SSPs to each other, on the other hand, helps to apply RV.

3. Mixing: Before the SP's can check the validity of identification tokens, they have to ensure that no coercer learns whether his token is valid or not (remember that a coercer knows where his ballot is residing on BB). Hence, the SP's jointly apply a verifiable mix to the ballots, which only consist of three public-key ciphertexts each, thus enabling efficient verifiable mixes. For brevity, we abuse notation in Figure 4.21 for lifting the verifiable mix to ballots.

4. Checking token validity: The initial ballot list is now disconnected from the remaining ballots such that we can proceed by excluding ballots submitted with fake tokens, i.e., those posted by coercers, enforcing condition four in Definition 4.9. Thanks to our ballot design, this check is a local property, checkable on every ballot individually without the need of auxiliary information, as required by classical approaches with quadratic complexity [55]. Concretely, the SP's recompute the token from the encrypted pseudonym $\hat{p}_i$ and check their equality under encryption via PET.

5. Decrypting the votes: Finally, enforcing condition three of Definition 4.9, the SP's decrypt the ratings.

The final result of Tally is the collection V of ratings v_i along with all the proofs and intermediate results obtained in this protocol, summarized in the variable π.

4.4.4. Discussion

The goal of this work is to design a privacy-preserving, publicly verifiable, coercion-resistant rating platform, with a single, albeit distributed, service provider. Although we leave a formal treatment of the generalization to multiple, coexisting rating services, and even further to fully-fledged *coercion-resistant ABCs*, as future work, we find it interesting to discuss the challenges involved in such a generalization along with possible solutions.

Attribute-hiding. Had we only a single SP, attribute-hiding would unconditionally hold for those attributes not required by SP (those specified in I). Instead, if we are confronted with several different SP's, each of those might come with a, possibly different, set I of required attributes. This extension opens a new attack angle: a bogus SP could request sensitive attributes and coerce the user so as to use that service. We thus envision a certified auditing procedure for SP's, which allows them to require only those attributes that are indeed necessary for their functionality.

Coercion-resistance. In e-voting, it is reasonable to require that the registration for both CI and SP have to be finished before the user can be coerced. This is instead problematic if the system is open and multiple rating services coexist. The solution that we have in mind lowers the level of coercion that is tolerated by the system. In CR^{2}P, as soon as a user is coerced, she can be successfully impersonated in all rating services that she is not registered with (due to the registration assumption). In this case, since we target rate buying and forced attribute disclosure, we let users announce to the PKI that their identity is lost. The PKI in turn puts the blinded identity on a black list of revoked identities. This list is then checked by SP's whenever a user wants to register. If that user is on the list, registration is aborted, at least by honest SP's, one of which always exists. Clearly, such a solution no longer protects against coercion by means of physical threat since the user publicly announces it.

Services without public output. While rating platforms and e-voting both have a public, verifiable output, this is not the case in general for arbitrary services. Achieving coercion-resistance in such a setting, for instance in a social network, is much more challenging, since it requires the design of fake, but yet plausible, web content.

4.5. Formal Results

We report the formal results of our cryptographic construction and prove them in Appendix C.3.

Theorem 4.1 (Attribute-hiding)**.** *Let* $\Pi_{\mathsf{PKE}}{}^{+}$ *be IND-CPA secure and allow for distributed key-generation and decryption,* Π_{SRC} *be message-hiding, and* ZKP *be a zero-knowledge proof system. Then* CR^{2}P *is attribute-hiding.*

Theorem 4.2 (Anonymity)**.** *Let* ZKP *be a zero-knowledge proof system and* SSP *be anonymous. Then, in the random oracle model,* CR^{2}P *achieves anonymity.*

Theorem 4.3 (Coercion-resistance)**.** *Let* Π_{PKE} *be IND-CPA secure and allow for distributed key-generation and decryption,* ZKP *be a zero-knowledge proof system, and assume the hardness of DDH. Then, in the random oracle model,* CR^{2}P *achieves coercion-resistance.*

Theorem 4.4 (End-to-end verifiability)**.** *Let* Π_{SRC} *be EUF-CMA,* SSP*'s be unique, and* ZKP *be a zero-knowledge proof system. Then* CR^{2}P *is end-to-end verifiable.*

4.6. Implementation & Experiments

We present the cryptographic realization of SRC (Section 4.6.1) and then summarize our experimental evaluation (Section 4.6.2). We list the cryptographic primitives that we build on in Table 4.3, while we postpone details and security proofs to Appendix C.2.

Scheme	Instantiation
BGSetup	MNT curves [147]
$\Pi_{\mathsf{PKE}}/\Pi_{\mathsf{PKE}}{}^{+}$	ElGamal [76] / Paillier [153]
Π_{SRC}	Section 4.6.1
Verifiable Mix nets	Shuffle proofs [22], RPC [110]
NIZKs via Fiat-Shamir heuristic [78]	DLog [173], EqDLog [40,59], PET [109],OR proof [58], Proof of plaintext knowledge [55,93], DVP [111]

Table 4.3.: Cryptographic scheme instantiations.

```
$(sk, vk) \leftarrow \mathsf{setup}_{\mathsf{PS}}(1^\lambda, \ell)$

$(x, y_1, \dots, y_\ell) \leftarrow_\$ \mathbb{Z}_q^{*\ell+1}$
$X = h^x$
for $1 \le i \le \ell$ do
    $Y_i = h^{y_i}$
$sk = (x, y_1, \dots, y_\ell)$
$vk = (h, X, Y_1, \dots, Y_\ell)$
return $(sk, vk)$
```

```
$\sigma \leftarrow \mathsf{sign}_{\mathsf{PS}}(sk, \vec{m})$

$a \leftarrow_\$ \mathbb{G}_1$
return $\sigma = (a, a^{x+\sum_{i=1}^{\ell} y_i m_i})$

$\{\top, \bot\} \leftarrow \mathsf{vfy}_{\mathsf{PS}}(vk, \sigma, \vec{m})$

if $\sigma_1 \ne 1_{\mathbb{G}_1}$ then    // $\sigma = (\sigma_1, \sigma_2)$
    if $e(\sigma_1, X \cdot \prod_{i=1}^{\ell} Y_i^{m_i}) = e(\sigma_2, h)$ then
        return $\top$
return $\bot$
```

Figure 4.22.: The PS signature scheme [160].

4.6.1. Instantiation of SRC

Our instantiation of SRC is an extension of the Pointcheval-Sanders (PS) signature scheme [160] (Figure 4.22), which works in the elliptic curve setting with a bilinear map. Let $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathsf{BGSetup}(1^\lambda)$ where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are groups over a prime-order finite field $\mathbb{F}_q$, $\langle g \rangle = \mathbb{G}_1$, $\langle h \rangle = \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map, i.e., (1) e is bilinear, i.e., for all $x, y \in \mathbb{F}_q$ we have $e(g^x, h^y) = e(g, h^y)^x = e(g^x, h)^y$, (2) e is non-degenerate, i.e., $\langle e(g, h) \rangle = \mathbb{G}_T$, and (3) e is efficiently computable. Moreover, we assume that BGSetup generates parameters for a type-III setup, i.e., there is no efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

SRC. Our construction is an extension of the PS signature scheme, detailed in Figure 4.23. Therein, we also show a way to sign blinded identities that can be verified using the unblinded identity.

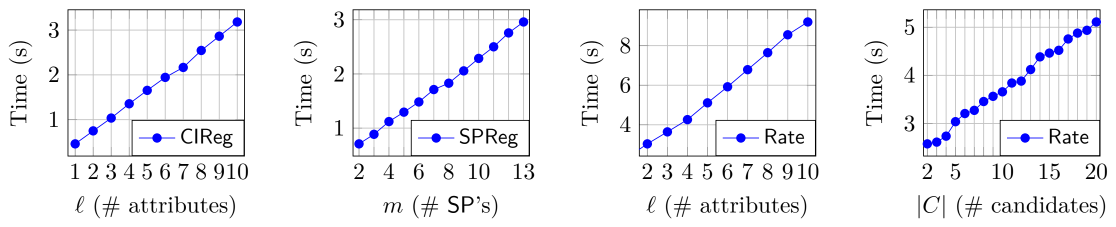
Security Results. We report the security guarantees of our construction in the following. The security proofs can be found in Appendix C.1.

Theorem 4.5 (EUF-CMA). *If the Pointcheval-Sanders signature scheme is EUF-CMA, so is SRC.*

Theorem 4.6 (Hiding). *SRC is message-hiding.*

```
$(sk, vk) \leftarrow \mathsf{setup}_{\mathsf{SRC}}(1^\lambda, n)$
    return $\mathsf{setup}_{\mathsf{PS}}(1^\lambda, n+1)$

$\sigma \leftarrow \mathsf{sign}(sk, \vec{m})$
    return $\mathsf{sign}_{\mathsf{PS}}((sk)_{\{1,\ldots,n\}}, \vec{m})$

$(\sigma, oi, aux) \leftarrow \mathsf{sign}_{\mathsf{C}}(sk, \vec{m}, r)$
    $a \leftarrow\!\!\$\ \mathbb{G}_1$
    $\sigma = (a, a^{x+(\sum_{i=1}^{n} y_i m_i)+r y_{n+1}})$
    return $(\sigma, r, a^{y_{n+1}})$

$(\sigma', oi, aux) \leftarrow \mathsf{comm}(sk, \sigma, r)$
    $(\sigma_1, \sigma_2) \leftarrow \sigma$
    $\sigma' = (\sigma_1, \sigma_2 \sigma_1^{r y_{n+1}})$
    return $(\sigma', r, \sigma_1^{y_{n+1}})$

$\sigma' \leftarrow \mathsf{rand}_{\mathsf{S}}(\sigma, vk, r)$
    $(\sigma_1, \sigma_2) \leftarrow \sigma$
    return $(\sigma_1^r, \sigma_2^r)$

$(\sigma', aux') \leftarrow \mathsf{rand}_{\mathsf{S}}(\sigma, aux, vk, r)$
    $(\sigma_1, \sigma_2) \leftarrow \sigma$
    return $((\sigma_1^r, \sigma_2^r), aux^r)$

$\sigma' \leftarrow \mathsf{rand}_{\mathsf{C}}(\sigma, aux, vk, s)$
    $(\sigma_1, \sigma_2) \leftarrow \sigma$
    return $(\sigma_1, \sigma_2 \cdot aux^s)$

$\{\top, \bot\} \leftarrow \mathsf{vfy}(vk, \sigma, \vec{m})$
    return $\mathsf{vfy}_{\mathsf{PS}}((vk)_{\{1,\ldots,n\}}, \vec{m})$

$\{\top, \bot\} \leftarrow \mathsf{vfy}_{\mathsf{C}}(vk, \sigma, \vec{m}, oi)$
    if $\sigma_1 \neq 1_{\mathbb{G}_1}$    // $\sigma = (\sigma_1, \sigma_2)$
        $t = X \cdot (\prod_{i=1}^{n} Y_i^{m_i} \cdot Y_{n+1}^{oi})$
        if $e(\sigma_1, t) = e(\sigma_2, h)$
            return $\top$
    return $\bot$

$\hat{\sigma} \leftarrow \mathsf{vfy}_{\mathsf{start}}(\sigma, vk, \vec{m}|_R)$
    $(\sigma_1, \sigma_2) \leftarrow \sigma$
    $t = X^{-1} \cdot \prod_{i \in R} Y_i^{-m_i}$
    $\sigma' = e(\sigma_2, h) \cdot e(\sigma_1, t)$
    return $\hat{\sigma} = (\sigma', \sigma_1)$

$\{\top, \bot\} \leftarrow \mathsf{vfy}_{\mathsf{fin}}(\hat{\sigma}, vk, \vec{m}|_{\bar{R}}, oi)$
    $(\hat{\sigma}_1, \hat{\sigma}_2) = \hat{\sigma}$
    $t_1 = e(\hat{\sigma}_2, Y_{n+1})^{oi}$
    $t_2 = e(\hat{\sigma}_2, \prod_{i \in \bar{R}} Y_i^{m_i})$
    if $\hat{\sigma}_1 = t_1 \cdot t_2$
        return $\top$
    return $\bot$

$\sigma = (a, a^{x+y_1 id})$

Blind signing of $id \in \mathbb{Z}_q$
    $bid \leftarrow g^{id}$
    $\alpha \leftarrow\!\!\$\ \mathbb{Z}_q^*$;    $a \leftarrow g^\alpha$
    return $\sigma = (a, a^x bid^{y_1 \alpha})$
```

Figure 4.23.: Our instantiation of SRC.



(a) CIReg time per user as $\ell = |\vec{at}|$ grows. (b) SPReg time per user as m grows. (c) Rate time per user as ℓ grows. (d) Rate time per user as $|C|$ grows.

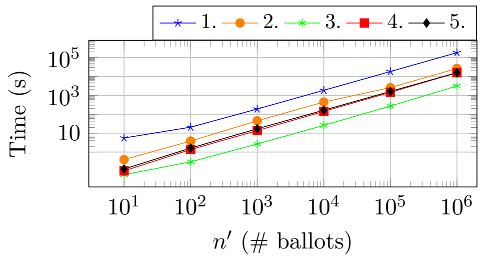
Figure 4.24.: Computation times for CIReg, SPReg, and Rate.

4.6.2. Experimental Results

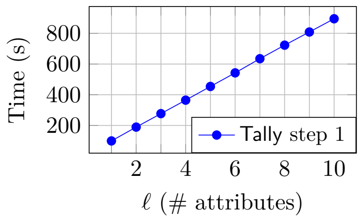
We implemented the algorithms described in Table 4.3 and Section 4.4 in Java. We instantiate all previously described schemes for 112 bits of security, which is considered to be secure until 2030 [32]. We ran a single rating process on an Intel Xeon (32 cores with hyper-threading, 2.6GHz each). We vary the number of attributes ℓ from 1 to 10, the number of service providers m from 2 to 13, the number of possible rating choices $|C|$ from 2 to 20, and the number of users n from 10 to 1M (in exponential steps). N_1 refers to the initial number of ballots on BB. The results are summarized in Figure 4.24 and Figure 4.25.

Computation. The CIReg protocol is practical, as depicted in Figure 4.24a; it ranges on average from 460 ms for one attribute up to 3.2 s for ten attributes. The SPReg protocol depends on the number of service providers: it lies in the range of around 700 ms up to 2.9 s (see Figure 4.24b). The time for Rate depends on the number of attributes necessary to access the service and the number of rating choices; varying the former, it ranges from 2.3 s to 9.2 s (see Figure 4.24c) while varying the latter, it ranges from 2.5 s to 5.1 s (see Figure 4.24d).

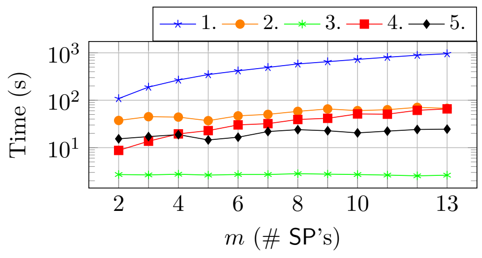
The most time consuming protocol is Tally, both generation and verification, which happen at once since service providers do not trust each other. We provide detailed measurements for each partial step in Figure 4.25a–4.25b. We show numbers only for randomized partial checking (RPC) since our implementation of proofs of shuffle correctness (PSC) [22] is at least two to three orders of magnitude slower than the RPC solution. It is not surprising that we do not achieve the computation times achieved by [22], since they implement several optimizations that go beyond the scope of this performance evaluation. But even with the optimizations in place, the RPC solution is still better in terms of computation, albeit not with respect to robustness since it requires an honest majority of service providers. Notice that robustness only affects coercion-resistance, for verifiability, all service providers may be malicious. Tally scales linearly in the number of service providers and users, with the first step, i.e., the ballot verification, being the by far most expensive step. This is not surprising given the complexity of the underlying protocol. When increasing the number of users, we get tally computation and verification times of roughly 6 hours for 100,000 users, which is more than enough for e-health rating systems



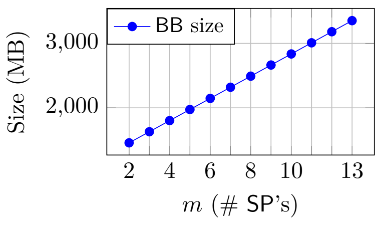
(a) Tally time with RPC as n' grows and $m = 3$, $\ell = 2$.



(b) Proof verification time (step 1 of Tally) as ℓ grows, $m = 3$, $n' = 1000$.



(c) Tally time with RPC as m grows and $n' = 1000$, $\ell = 2$.



(d) BB size as m grows, $n' = 100000$, $\ell = 2$, and $|C| = 5$.

Figure 4.25.: Computation times for Tally and sizes for BB.

and would even be sufficient for small-size city elections, or when distributing elections to smaller regions, which is what usually happens in state-wide elections.

BB size. Finally, the size of the initial BB is determined by the number of ballots, the size of which is 8.5 KB plus roughly 1 KB per attribute and plus about 0.8 KB per additional rating choice. To assess the size of the overall BB including proofs of SP's, we depict in Figure 4.25d the size of BB for varying SP, 1000 initial ballots, five rating choices, and two attributes. This curve indicates the amount of data an external verifier would have to download so as to verify the rating process.

Communication. The ClReg protocol requires to exchange $3.5 + 14\ell$ KB for the credentials and the proof of correctness. In SPReg, the user communicates with the service providers around 3.7 KB per service provider. In Rate, the user has to send the ballot to BB, the size of which we explained in the previous paragraph. The communication cost is thus moderate, despite the protocol complexity.

4.7. Related Work

Privacy-preserving reputation systems. There is a long line of research on privacy-preserving reputation systems [6, 31, 97, 114, 119, 128, 148, 157, 158, 171, 190]. Most of these schemes do not guarantee coercion-resistance, either because they deploy standard

ABCs [6,31,148,158] or because the system design allows an adversary to corrupt parties completely and to act on their behalf [97,119,128,171]. Some achieve privacy by assuming the existence of a trusted third party [157,190], an assumption which might be helpful to achieve coercion-resistance, but the mentioned works do not address that.

Finally, Kerschbaum [114] presents a reputation scheme that aims at coercion-resistance. The idea is to use two mutually distrustful servers to let one party rate the other. Besides the setup, a difference between this scheme and the others, including ours, is that it hides the individual user rating rather than the link between the user and its rating. A more precise and formal comparison with that work is difficult, since coercion-resistance is neither formalized nor proved.

Attribute-based credentials. ABCs have been widely studied in the literature and still constitute an active subject of research [14,18,23,24,39,74,84,105,143]. Nevertheless, as we illustrate in this chapter, previous ABCs fall short of providing adequate security and privacy guarantees in case of coercion. The fundamental problem with standard ABCs is that they reveal the attributes and knowing the credential suffices to impersonate the user. Our design of SRC, in contrast, hides the signed attributes and does not allow for proving their knowledge unless one knows the opening information of the commitment, which allows us to achieve coercion-resistance. In CR^{2}P, the opening information is encrypted for the service provider by the credential issuer and the user never learns it. Moreover, using designated verifier proofs, one can prove the correctness of the attributes signed in a SRC to a dedicated person, which is crucial for the correctness of CR^{2}P, without leaking any information to any third-party, including the coercer.

Electronic voting. A formal treatment of coercion-resistance has been developed in the context of e-voting, both for paper-based schemes [33,46,48,124,169] and for remote e-voting systems like the one of Juels *et al.* [112] (JCJ), Civitas [55], and a few more (see [10–12,54,83,172,181,182] and [21,102] for a weaker notion called receipt-freeness). Additionally, the community has proposed frameworks, models, and formal (symbolic or cryptographic) definitions for coercion resistance so as to formally analyze existing e-voting protocols [16,68,100,125]. CR^{2}P differs from this line of work in three fundamental aspects. First, coercion-resistance is achieved in a stronger attacker model, with a single malicious credential issuer as opposed to distributed ones out of which one is assumed to be honest. Secondly, CR^{2}P does not rely on any communication between the issuer and the service provider, while in remote e-voting the registrars typically pass information to the talliers through a shared bulletin board. Finally, the tallying procedure is linear in the size of ratings, while in several remote e-voting schemes it is quadratic [55,112].

Some other works aim at downsizing the quadratic tallying complexity of Civitas and JCJ to linear complexity. A first approach [54,172,181,182] relies a technique introduced by Weber *et al.* [191] for duplicate elimination based on blinded credential decryption. The complexity of the credential validity check is still quadratic, but the efficiency can be customized by means of anonymity sets of size β, which reduce the complexity to $O(\beta N)$ [54,172] and $O(\beta n + N)$ [181,182], respectively, where N is the number of ballots and n is the number of registered voters. Our protocol does not need to apply blinding before decryption, since we rely on pseudonyms that are unlinkable to the user. A different approach is proposed by Araújo *et al.* [10–12], and it is based on equality checks of values that are linkable to the voter and which can be faked in case of coercion. This, along with ours, is the only approach for checking the validity of credentials that is truly linear. To check that a credential is valid, the registrars recompute, for every ballot, the signature [11, 12] (resp. MAC [10]) under encryption. If the recomputed version is equivalent to the one submitted in the ballot, it is considered valid. The key difference from our approach is that we do not require the registrar (credential issuer) in the tallying phase, which is crucial for rating systems, since the tallying process must be independent of the credential issuer.

5. Conclusion and Outlook

Digitization is often considered a revolution since it changes many areas of individual life. It affects businesses and people alike. One major challenge of the digitization is the privacy of personal and, in particular, sensitive user data. At the same time, that data shall be usable for authorized personnel for necessary actions. For instance, a necessary action could be the storage of the results of a medical treatment or the usage of medical information for the purpose of research while keeping individuals anonymous. Furthermore, digitization does not only have to protect the user from others but also the user from himself so as to not to reveal too much about herself in certain cases. Such a case could be the expression of a negative opinion about a person in a rating platform, which should not deanonymize the user.

In this thesis, we presented cryptographic techniques for three different domains of the digitization that we conclude in the sequel.

Privacy and access control for outsourced storage. In the first part of this thesis, we introduced the concept of Group ORAM, which captures an unprecedented range of security and privacy properties in the cloud storage setting. Based on our definitional framework Π_{GORAM}, we establish a lower bound on the server-side computational complexity, showing that any Group ORAM that is oblivious against malicious clients has to involve at least $\Omega(n)$ computation steps. We further present a novel cryptographic instantiation, which achieves an amortized communication overhead of $O(\sqrt{n})$ by combining private information retrieval technologies, a new accumulation technique, and an oblivious gossiping protocol. Access control is enforced by integrity proofs. Finally, we showed how to bypass our lower bound by leveraging a trusted proxy [170], thereby achieving logarithmic communication and server side computational complexity. We then move to Group ORAM that is oblivious only against the server: the fundamental idea underlying our instantiation is to extend a state-of-the-art ORAM scheme [187] with access control mechanisms and integrity proofs while preserving obliviousness. To tackle the challenge of devising an efficient and scalable construction, we devised two novel zero-knowledge proof techniques for shuffle correctness as well as a new accountability technique based on chameleon signatures, both of which are generically applicable and of independent interest. We showed how Π_{GORAM} is an ideal solution for personal record management systems.

Secure querying of outsourced databases. In the second part of this thesis, we contribute to the field of order-preserving encryption (OPE), which is an enabling technology to implement database applications on encrypted data: the idea is that the ordering of ciphertexts matches the one of plaintexts so that inequalities on encrypted data are efficiently computable. Recent works, however, showed that various attacks can be mounted by exploiting the inherent leakage of plaintext frequency. Frequency-hiding OPE [115] is a stronger primitive that aims at solving this problem by hiding the frequency of plaintexts. Unfortunately, as we show, the definition of frequency-hiding OPE is imprecise and a natural interpretation of it leads to concrete attacks against the construction presented in [115]. We find and identify the problem, which is related to the imprecision in the definition and which corresponds to that attack, in the security proof. We then formulate a more general impossibility result, proving that the security definition introduced in [115] cannot be achieved by any OPE scheme. In order to complete the picture and assess which theoretical security is achievable at all, we make the definition in [115] more precise by giving the challenger more capabilities and augmenting the OPE model so as to receive randomized orders as inputs which are used to break ties. We finally show that the more precise version of the definition can be achieved by a variant of the construction introduced in [115].

Coercion-resistant rating platforms. Finally, in the third part of this thesis, we presented CR^{2}P, a coercion-resistant rating platform. Rating platforms are a central asset in marketing strategies and are one of the main consultations for individual decision making. As a matter of fact, they do not provide sufficient security and privacy guarantees to protect against fraud in terms of fake ratings and suppressed ratings, for short, coercive attacks. To counteract those attacks, CR^{2}P achieves coercion-resistance, while retaining strong access control guarantees through attribute-based credentials, anonymity and attribute-hiding. We have formalized the salient privacy properties in this context, presented a provably secure cryptographic realization, and demonstrated its feasibility for even large-size rating situations. The most significant technical novelties of our construction are signatures on randomized commitments and identification tokens. The former are a novel kind of attribute-based credentials, which allow the user to prove the knowledge of a subset of attributes while leaving the rest of the verification and the opening of the remaining attributes to the verifier, in our case the rating platform. The latter are our key ingredient to achieve coercion-resistance: they can be faked in case of coercion without the coercer being able to notice that. Furthermore, they are bound to the user's identity and facilitate the verification of the rating process, which enables a linear tallying procedure.

Outlook. This thesis opens up a number of interesting research directions.

In the area of multi-client ORAM, it would be interesting to complete the picture by a lower bound on the communication complexity. Furthermore, we would like to relax the obliviousness property in order to bypass the computational lower bound, coming up with more efficient constructions and quantifying the associated privacy loss. A further research goal is the design of cryptographic solutions allowing for applications on outsourced storage, for instance, an tool that allows clients to learn only limited information (e.g., statistics) about the dataset.

Concerning our investigations and findings in the area of OPE, it must be said that despite this seemingly positive results, in the presence of the plethora of empirical attacks against (FH-)OPE and its variants (e.g., ORE), we suggest to not use any of those schemes for actual deployment since the security guarantees achieved do not reflect practical requirements. We recommend to move away from OPE in general, more towards other alternatives, even if there are none that solve the problem so conveniently; at the price of low to no security.

In the area of privacy-preserving rating platforms, it would be interesting to investigate how far the topic of coercion can be pushed to other areas and scenarios. We believe that coercion might also be problematic in other situations even though this has not been considered yet and required solutions are, hence, missing. Moreover, we would like to boost the efficiency of our construction so as to make it amenable to large-scale systems with millions of ratings that are to be processed in short periods of time. Finally, it would be interesting to investigate whether we can lift coercion-resistance to attribute-based credential systems in general.

Appendices

A. Cryptographic Building Blocks and Definitions

A.1. Private-Key Encryption

We recall the basic definition of private-key encryption and the respective IND-CPA security definition [89].

Definition A.1 (Private-Key Encryption)**.** *A* private key encryption scheme *is a tuple of* PPT *algorithms* $\Pi_{\mathsf{SE}} = (\mathsf{Gen}_{\mathsf{SE}}, \mathcal{E}, \mathcal{D})$*, where the generation algorithm* $\mathsf{Gen}_{\mathsf{SE}}(1^\lambda)$ *outputs a private key k; the encryption algorithm* $\mathcal{E}(k, m)$ *takes as input a key k and a message* $m \in \mathcal{M}$ *and outputs a ciphertext c; the decryption algorithm* $\mathcal{D}(k, c)$ *takes as input a key k and a ciphertext c and outputs a message m.*

A private key encryption scheme is *correct* if and only if, for all $k \leftarrow \mathsf{Gen}_{\mathsf{SE}}(1^\lambda)$ and all messages $m \in \mathcal{M}$ we have $\mathcal{D}(k, \mathcal{E}(k, m)) = m$.

Next, we define IND-CPA security for private key encryption schemes, where $\mathcal{O}_k(\cdot)$ is an encryption oracle that returns $\mathcal{E}(k, m)$ when queried on a message m.

Definition A.2 (IND-CPA Security)**.** *Let* $\Pi_{\mathsf{SE}} = (\mathsf{Gen}_{\mathsf{SE}}, \mathcal{E}, \mathcal{D})$ *be a private key encryption scheme.* Π_{SE} *has* indistinguishable ciphertexts against chosen-plaintext attacks *if for all* PPT *adversaries* $\mathcal{A}$ *the following probability is negligible (as function in* λ*):*

$$\left|\mathsf{Pr}\left[\mathsf{Exp}^{\Pi_{\mathsf{SE}}}_{\mathcal{A},\mathsf{cpa}}(\lambda, 1) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\Pi_{\mathsf{SE}}}_{\mathcal{A},\mathsf{cpa}}(\lambda, 0) = 1\right]\right|$$

where $\mathsf{Exp}^{\Pi_{\mathsf{SE}}}_{\mathcal{A},\mathsf{cpa}}(\lambda, b)$ *is the following experiment:*

```
$\mathsf{Exp}^{\Pi_{\mathsf{SE}}}_{\mathcal{A},\mathsf{cpa}}(\lambda, b)$
$k \leftarrow \mathsf{Gen}_{\mathsf{SE}}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_k(\cdot)}$
$c_b \leftarrow \mathcal{E}(k, m_b)$
$b' \leftarrow \mathcal{A}(c_b)$
if $|m_0| = |m_1| \wedge b' = b$ then
    return 1
return 0
```

Finally, we introduce the notion of elusive-range scheme, we denote the range of a key k by $\mathsf{Range}_\lambda(k) := \{\mathcal{E}(k, m)\}_{m \in \{0,1\}^\lambda}$.

Definition A.3 (Elusive Range [134]). *Let* $\Pi_{\mathsf{SE}} = (\mathsf{Gen}_{\mathsf{SE}}, \mathcal{E}, \mathcal{D})$ *be a private key encryption scheme.* Π_{SE} *has* elusive-range *if* $\mathsf{Pr}_{k \leftarrow \mathsf{Gen}_{\mathsf{SE}}(1^\lambda)}[\mathcal{A}(1^\lambda) \in \mathsf{Range}_\lambda(k)]$ *is negligible in* λ *for all* PPT *adversaries* $\mathcal{A}$*.*

A.2. Public-Key Encryption

We recall the basic definition of public-key encryption [70] and the corresponding IND-CPA [89] and IND-CCA [60] security definitions.

Definition A.4 (Public Key Encryption). *A* public key encryption scheme *is a tuple of* PPT *algorithms* $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{E}, \mathsf{D})$*, where the generation algorithm* $\mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$ *outputs an encryption key ek and a decryption key dk; the encryption algorithm* $\mathsf{E}(ek, m)$ *takes as input the encryption key ek and a message* $m \in \mathcal{M}$ *and outputs a ciphertext c; the decryption algorithm* $\mathsf{D}(dk, c)$ *takes as input the decryption key dk and a ciphertext c and outputs a message m or* $\perp$*.*

A public key encryption scheme is *correct* if and only if, for all $(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$ and all messages $m \in \mathcal{M}$ we have $\mathsf{D}(dk, \mathsf{E}(ek, m)) = m$.

Next, we define IND-CPA security for public-key encryption schemes.

Definition A.5 (CPA Security). *Let* $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{E}, \mathsf{D})$ *be a public key encryption scheme.* Π_{PKE} *has* indistinguishable ciphertexts against chosen-plaintext attacks *(CPA) if for all* PPT *adversaries* $\mathcal{A}$ *the following probability is negligible (in the security parameter* λ*):*

$$\left| \mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A},\mathsf{cpa}}^{\Pi_{\mathsf{PKE}}}(\lambda, 1) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A},\mathsf{cpa}}^{\Pi_{\mathsf{PKE}}}(\lambda, 0) = 1\right] \right|$$

where $\mathsf{Exp}_{\mathcal{A},\mathsf{cpa}}^{\Pi_{\mathsf{PKE}}}(\lambda, b)$ *is the following experiment:*

```
$\mathsf{Exp}_{\mathcal{A},\mathsf{cpa}}^{\Pi_{\mathsf{PKE}}}(\lambda, b)$
$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}(ek)$
$c_b \leftarrow \mathsf{E}(ek, m_b)$
$b' \leftarrow \mathcal{A}(c_b)$
if $|m_0| = |m_1| \wedge b' = b$ then
    return 1
return 0
```

For our purposes, we need an IND-CPA-secure public-key encryption scheme that is rerandomizable. Hence, we assume that there exists a function $\mathsf{Rnd}(ek, c, r)$ that takes as input an encryption key ek, a ciphertext c, and randomness r and returns a rerandomized ciphertext c' encrypting the same content where $c \neq c'$ and both c and c' have the same distribution in the ciphertext space. Security is given if no adversary can distinguish whether c' is the re-randomization of c or an encryption of a different message.

Finally, we report the definition of IND-CCA security. Let $\mathcal{O}_{dk}(\cdot)$ be an oracle that, on input a ciphertext c outputs $\mathsf{D}(dk, c)$.

Definition A.6 (CCA Security). *Let* $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{E}, \mathsf{D})$ *be a public key encryption scheme.* Π_{PKE} *has* indistinguishable ciphertexts against chosen-ciphertext attacks *(CCA) if for all* PPT *adversaries* $\mathcal{A}$ *the following probability is negligible (in the security parameter* λ*):*

$$\left|\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{PKE}}}_{\mathcal{A},\mathsf{cca}}(\lambda, 1) = 1\right] - \Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{PKE}}}_{\mathcal{A},\mathsf{cca}}(\lambda, 0) = 1\right]\right|$$

where $\mathsf{Exp}^{\Pi_{\mathsf{PKE}}}_{\mathcal{A},\mathsf{cca}}(\lambda, b)$ *is the following experiment:*

```
$\mathsf{Exp}^{\Pi_{\mathsf{PKE}}}_{\mathcal{A},\mathsf{cca}}(\lambda, b)$
$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{dk}(\cdot)}(ek)$
$c_b \leftarrow \mathsf{E}(ek, m_b)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}'_{dk}(\cdot)}(c_b)$   // $\mathcal{O}'_{dk}(\cdot)$ aborts on input $c_b$
if $|m_0| = |m_1| \wedge b' = b$ then
    return 1
return 0
```

A.3. Predicate Encryption

We recall the notion of predicate encryption [113]. In a predicate encryption scheme one can encrypt a message m under a certain attribute $I \in \Sigma$ using a master public key ppk where Σ is the universe of all possible attributes. Furthermore, one can decrypt the resulting ciphertext using a secret key psk_f associated with a predicate $f \in \mathcal{F}$ if and only if I fulfills f, i.e., $f(I) = 1$, where $\mathcal{F}$ is the universe of all predicates.

Definition A.7 (Predicate Encryption). *A* predicate encryption scheme *for the universe of predicates and attributes* $\mathcal{F}$ *and* Σ*, respectively, is a tuple of* PPT *algorithms* $\Pi_{\mathsf{PE}} = (\mathsf{Gen}_{\mathsf{PE}}, \mathsf{K}_{\mathsf{PE}}, \mathsf{E}_{\mathsf{PE}}, \mathsf{D}_{\mathsf{PE}})$*, where the generation algorithm* $\mathsf{Gen}_{\mathsf{PE}}$ *takes as input a security parameter* 1^λ *and returns a master public and a master secret key pair* $(ppk, pmsk)$*; the key generation algorithm* K_{PE} *takes as input the master secret key pmsk and a predicate description* $f \in \mathcal{F}$ *and returns a secret key* psk_f *associated with* f*; the encryption algorithm* E_{PE} *takes as input the master public key ppk, an attribute* $I \in \Sigma$*, and a message* m *and it returns a ciphertext c; and the decryption algorithm* D_{PE} *takes as input a secret key* psk_f *associated with a predicate* f *and a ciphertext* c *and outputs either a message* m *or* $\perp$*.*

A predicate encryption scheme Π_{PE} is *correct* if and only if, for all λ, all key pairs $(ppk, pmsk) \leftarrow \mathsf{Gen}_{\mathsf{PE}}(1^\lambda)$, all predicates $f \in \mathcal{F}$, all secret keys $psk_f \leftarrow \mathsf{K}_{\mathsf{PE}}(pmsk, f)$, and all attributes $I \in \Sigma$ we have that (i) if $f(I) = 1$ then $\mathsf{D}_{\mathsf{PE}}(psk_f, \mathsf{E}_{\mathsf{PE}}(ppk, I, m)) = m$ and (ii) if $f(I) = 0$ then $\mathsf{D}_{\mathsf{PE}}(psk_f, \mathsf{E}_{\mathsf{PE}}(ppk, I, m)) = \perp$ except with negligible probability in λ.

Next, we recall the security notion *attribute-hiding* that we require the predicate encryption scheme to hold. To give an intuition, suppose that there are professors, students, and employees at a university with corresponding attributes *Prof*, *Emp*, and *Stud*. Naturally, every member of a group will be equipped with a secret key psk_f such that f is either the predicate mayAccProf, mayAccEmp, or mayAccStud. We use the toy policy that professors may read everything and employees and students may read only encryptions created using *Emp* and *Stud*, respectively. Now, attribute-hiding means the following: let *file* be a file encrypted using the attribute *Prof*. On the one hand, a student equipped with $psk_{\mathsf{mayAccStud}}$ can neither decrypt the file nor tell with which attribute it is encrypted except for that it was not *Stud*. On the other hand, even a professor does not learn under which attribute *file* was encrypted, she only learns the content of the file and nothing more. The following definition formalizes the intuition given above where $\mathcal{O}_{pmsk}(\cdot)$ on input a predicate $f \in \mathcal{F}$ outputs psk_f if and only if $f(I_0) = f(I_1)$ and I_b are the challenge attributes of the adversary $\mathcal{A}$ and where $\mathcal{I}$ is the collection of attributes queried to $\mathcal{O}_{pmsk}(\cdot)$, which is filled upon every successful oracle call.

Definition A.8 (Attribute Hiding). *Let Π_{PE} be a predicate encryption scheme with respect to $\mathcal{F}$ and Σ. Π_{PE} is attribute hiding if for all* PPT *adversaries $\mathcal{A}$ the following probability is negligible:*

$$\left|\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{PE}}}_{\mathcal{A},\mathsf{ah}}(\lambda, 1) = 1\right] - \Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{PE}}}_{\mathcal{A},\mathsf{ah}}(\lambda, 0) = 1\right]\right|$$

where $\mathsf{Exp}^{\Pi_{\mathsf{PE}}}_{\mathcal{A},\mathsf{ah}}(\lambda, b)$ is the following experiment:

```
$\mathsf{Exp}^{\Pi_{\mathsf{PE}}}_{\mathcal{A},\mathsf{ah}}(\lambda, b)$
$\Sigma^2 \ni (I_0, I_1) \leftarrow \mathcal{A}(1^\lambda)$
$(ppk, pmsk) \leftarrow \mathsf{Gen}_{\mathsf{PE}}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{pmsk}(\cdot)}(ppk)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{pmsk}(\cdot)}(\mathsf{E}_{\mathsf{PE}}(ppk, I_b, m_b))$
if $b = b' \wedge |m_0| = |m_1| \wedge ((\exists f \in \mathcal{I}.\ f(I_0) = f(I_1) = 1) \implies m_0 = m_1)$ then
  return 1
return 0
```

As for public-key encryption, we require a rerandomization operation $\mathsf{R}_{\mathsf{PE}}(ppk, c, r)$.

We briefly describe below how to encode the access control matrix through predicates and attributes. We use $\Sigma = \mathcal{F} = \mathbb{Z}_q^{n+1}$ where n is the maximum number of clients that are registered with the database owner. Let $f, I \in \mathbb{Z}_q^{n+1}$ such that $f(I) = 1$ if and only if $\langle f, I \rangle = 0$, i.e., the two vectors f and I are orthogonal. Let $(f_1, \dots, f_n) \in \mathbb{Z}_q^{(n+1)\times n}$ be the matrix formed of all column-vectors representing the n clients. Let us furthermore assume that all the n columns are pairwise linearly independent. Now, in order to find an attribute that implements the read or write access modes of a data entry at index i for all clients, one computes a vector $I \in \mathbb{Z}_q^{n+1}$ that is orthogonal to the $k \leq n$ vectors corresponding to the clients that have access to i and that is not orthogonal to the other

$n - k$. Since there are at most n vectors to which I has to be orthogonal, there always exists a solution to this system of equations.

A.4. Broadcast Encryption

We recall the definition of broadcast encryption and an adaptive security notion [85].

Definition A.9 (Broadcast encryption)**.** *A* broadcast encryption scheme *is a tuple of* PPT *algorithms* $\Pi_{\mathsf{BE}} = (\mathsf{Gen}_{\mathsf{BE}}, \mathsf{K}_{\mathsf{BE}}, \mathsf{E}_{\mathsf{BE}}, \mathsf{D}_{\mathsf{BE}})$*:*

- *the generation algorithm* $\mathsf{Gen}_{\mathsf{BE}}(1^\lambda, n)$ *takes as input a security parameter* λ *and a maximum number of users* n *and outputs a key pair* (bsk, bpk)*;*

- *the key generation algorithm* $\mathsf{K}_{\mathsf{BE}}(i, bsk)$ *takes as input an index* $i \in \{1, \dots, n\}$ *and the secret key* bsk *and outputs a private key* d_i*;*

- *the encryption function* $\mathsf{E}_{\mathsf{BE}}(S, bpk)$ *takes as input a set* $S \subseteq \{1, \dots, n\}$ *and the public key* bpk *and outputs a pair* $\langle Hdr, K\rangle$ *where* Hdr *is called the header and* K *is called the message encryption key. Let* Π_{SE} *be a symmetric encryption scheme and let* $c \leftarrow \mathcal{E}(K, M)$ *be the encryption of message* M *that is supposed to be broadcast to users in* S*. Then the broadcast message consists of* (S, Hdr, c)*;*

- *finally, the decryption function* $\mathsf{D}_{\mathsf{BE}}(S, i, d_i, Hdr, bpk)$ *takes as input the set of users* S*, an index* i *with corresponding private key* d_i*, the header* Hdr*, and the public key* bpk*. If* $i \in S$ *and* d_i *belongs to* i*, then it outputs a message encryption key* K *that can be used to decrypt the symmetric-key ciphertext* c *produced during encryption.*

For the adaptive security definition below, we let $\mathcal{O}_{bsk}(\cdot)$ denote an oracle that on input an integer $1 \leq i \leq n$ outputs the corresponding secret key $d_i \leftarrow \mathsf{K}_{\mathsf{BE}}(i, bsk)$. The queried i's are collected by the oracle in a set $\mathcal{I}$. Furthermore, in the following formalization, we denote by $\mathcal{K}$ the key space of Π_{BE}.

Definition A.10 (Adaptive security)**.** *Let* Π_{BE} *be a broadcast encryption scheme.* Π_{BE} *is* adaptively secure *if for all* PPT *adversaries* $\mathcal{A}$ *the following probability is negligible:*

$$\left|\mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A}}^{\Pi_{\mathsf{BE}}}(\lambda, 1) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A}}^{\Pi_{\mathsf{BE}}}(\lambda, 0) = 1\right]\right|$$

where $\mathsf{Exp}_{\mathcal{A}}^{\Pi_{\mathsf{BE}}}(\lambda, b)$ *is the following experiment:*

```
$\mathsf{Exp}_{\mathcal{A}}^{\Pi_{\mathsf{BE}}}(\lambda, b)$
$(bsk, bpk) \leftarrow \mathsf{Gen}_{\mathsf{BE}}(1^{\lambda}, n)$
$S^* \leftarrow \mathcal{A}^{\mathcal{O}_{bsk}(\cdot)}(bpk)$
$\langle Hdr^*, K_0 \rangle \leftarrow \mathsf{E}_{\mathsf{BE}}(S^*, bpk)$
$K_1 \leftarrow_{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}(Hdr^*, K_b)$
**if** $b' = b \land S^* \cap \mathcal{I} = \emptyset$ **then**
  **return** $1$
**return** $0$
```

A.5. Chameleon Hash Functions

We recall the definition of chameleon hash functions as well as the notion of key-exposure freeness [13].

Definition A.11 (Chameleon hash function)**.** *A* chameleon hash function *is a tuple of* PPT *algorithms* $\Pi_{\mathsf{CH}} = (\mathsf{Gen}_{\mathsf{CHF}}, \mathsf{CH}, \mathsf{Col})$*, where the generation algorithm* $\mathsf{Gen}_{\mathsf{CHF}}(1^{\lambda})$ *outputs a key pair* (cpk, csk)*; the chameleon hash function* $\mathsf{CH}(cpk, m, r)$ *takes as input the public key cpk, a message m, and randomness r and it outputs a tag t; the collision function* $\mathsf{Col}(csk, m, r, m')$ *takes as input the private key csk, a message m, randomness r, and a new message* m' *and it outputs a new randomness* r' *such that* $\mathsf{CH}(cpk, m, r) = t = \mathsf{CH}(cpk, m', r')$*.*

We define next the key-exposure freeness property for chameleon hash functions [13] where $\mathcal{O}_{csk}(\cdot)$ is an oracle that on input a message m' outputs $r' \leftarrow \mathsf{Col}(csk, m, r, m')$ where (m, r) is initially chosen by the adversary. The oracle collects the collisions (m', r') in a set $\mathcal{I}$.

Definition A.12 (Key-exposure freeness)**.** *Let* $\Pi_{\mathsf{CH}} = (\mathsf{Gen}_{\mathsf{CHF}}, \mathsf{CH}, \mathsf{Col})$ *be a chameleon hash function.* Π_{CH} *is* key-exposure free *if* $\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathsf{kef}}^{\Pi_{\mathsf{CH}}}(\lambda) = 1\right]$ *is negligible in* λ *for all* PPT *adversaries* $\mathcal{A}$*, where* $\mathsf{Exp}_{\mathcal{A},\mathsf{kef}}^{\Pi_{\mathsf{CH}}}(\lambda)$ *is the following experiment:*

```
$\mathsf{Exp}_{\mathcal{A},\mathsf{kef}}^{\Pi_{\mathsf{CH}}}(\lambda)$
$(cpk, csk) \leftarrow \mathsf{Gen}_{\mathsf{CHF}}(\lambda)$
$(m, r) \leftarrow \mathcal{A}(cpk)$
$(m^*, r^*) \leftarrow \mathcal{A}^{\mathcal{O}_{csk}(\cdot)}(cpk)$
**if** $\mathsf{CH}(cpk, m, r) = \mathsf{CH}(cpk, m^*, r^*) \land \forall (m', r') \in \mathcal{I}.\ (m^*, r^*) \neq (m', r')$ **then**
  **return** $1$
**return** $0$
```

A.6. Digital Signatures

We recall the definition of digital signatures as well as the one of existential unforgeability [90].

Definition A.13 (Digital signature)**.** *A* digital signature scheme *is a tuple of* PPT *algorithms* $\Pi_{\mathsf{DS}} = (\mathsf{Gen}_{\mathsf{DS}}, \mathsf{sign}, \mathsf{vfy})$ *where the generation algorithm* $\mathsf{Gen}_{\mathsf{DS}}(1^\lambda)$ *outputs a key pair* (vk, sk)*; the signing function* $\mathsf{sign}(sk, m)$ *takes as input the signing key sk and a message m and it outputs a signature* σ*; the verification function* $\mathsf{vfy}(vk, \sigma, m)$ *takes as input the verification key vk, a signature* σ*, and a message m, and it outputs either* $\top$ *if* σ *is a valid signature for m or* $\top$ *otherwise.*

We next define existential unforgeability [90] where $\mathcal{O}_{sk}(\cdot)$ is an oracle that on input a message m outputs $\sigma \leftarrow \mathsf{sign}(sk, m)$. The so constructed pairs (m, σ) are collected in a set $\mathcal{I}$ by the oracle.

Definition A.14 (Existential unforgeability)**.** *Let* $\Pi_{\mathsf{DS}} = (\mathsf{Gen}_{\mathsf{DS}}, \mathsf{sign}, \mathsf{vfy})$ *be a digital signature scheme.* Π_{DS} *is* existentially unforgeable against chosen message attacks *if* $\mathsf{Pr}\left[\mathsf{Exp}^{\Pi_{\mathsf{DS}}}_{\mathcal{A},\mathsf{euf}}(\lambda) = 1\right]$ *is negligible in* λ *for all* PPT *adversaries* $\mathcal{A}$*, where* $\mathsf{Exp}^{\Pi_{\mathsf{DS}}}_{\mathcal{A},\mathsf{euf}}(\lambda)$ *is the following experiment:*

```
$\mathsf{Exp}^{\Pi_{\mathsf{DS}}}_{\mathcal{A},\mathsf{euf}}(\lambda)$
$(vk, sk) \leftarrow \mathsf{Gen}_{\mathsf{DS}}(\lambda)$
$(\sigma^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}_{sk}(\cdot)}(vk)$
**if** $\mathsf{vfy}(vk, \sigma^*, m^*) = \top \wedge \forall (m, \sigma) \in \mathcal{I}.\ m^* \neq m$ **then**
  **return** $1$
**return** $0$
```

A.7. Service-Specific Pseudonyms

SSPs [143] enjoy two fundamental properties that we require for our instantiation of CR^{2}P, uniqueness and anonymity, which we recall below.

Definition A.15 (Uniqueness)**.** *A service-specific pseudonym is* unique *[143] whenever the following statements hold with overwhelming probability where h is a hash function mapping bit strings to an SSP compatible group:*

1. *for any two honestly generated identities* id_1, id_2 *and any service* $\mathcal{S}$ *we have* $\mathsf{SSP}(id_1, \mathcal{S}) \neq \mathsf{SSP}(id_2, \mathcal{S})$*;*
2. *for any identity id and any service* $\mathcal{S}$*,* $\mathsf{SSP}(id, \mathcal{S})$ *is a unique value;*
3. *finally, for any identity id and any two service descriptions* S_1 *and* S_2 *we have* $\mathsf{SSP}(id, h(S_1)) \neq \mathsf{SSP}(id, h(S_2))$*.*

Definition A.16 (Anonymity). *A set of k pseudonyms $\{psd_1, \ldots, psd_k\}$ for k services for the same identity id provides* anonymity *if and only if, given a set $\{bid_1, \ldots, bid_m\}$ or m blinded identities, any* PPT *adversary can associate which bid_i corresponds to the pseudonyms with probability at most $1/m + negl(\lambda)$.*

A.8. Zero-Knowledge Proofs

A *zero-knowledge proof system* ZKP is a proof system executed between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ for some statement $\exists \vec{x}.\ F(\vec{x}, \vec{y})$ in which we call $\vec{x}$ the witnesses of the statement and $\vec{y}$ the public components. Being a proof system, ZKP enjoys two fundamental properties: *correctness*, i.e., an honest prover and an honest verifier always succeed in proving a correct statement, and *soundness*, i.e., a malicious prover cannot convince an honest verifier of a false statement except with negligible probability. A zero-knowledge proof system additionally has the *zero-knowledge* property, i.e., the proof transcript generated between $\mathcal{P}$ and $\mathcal{V}$ does not reveal more than the validity of $F(\vec{x}, \vec{y})$. More formally, zero-knowledge is formalized by requiring the existence of a simulator $\mathcal{S}$ that on input F and $\vec{y}$, generates a verifying transcript that no malicious $\mathcal{V}$ can distinguish from an honestly produced transcript.

A zero-knowledge proof *of knowledge* additionally requires that the prover knows the witnesses $\vec{x}$. This is formalized by requiring the existence of an extractor $\mathcal{E}$ that, on input a proof transcript, extracts a correct witness with overwhelming probability.

Finally, a *non-interactive* zero-knowledge proof of knowledge, which we abbreviate throughout the paper with NIZK, is a proof that consists of a single message sent from the prover to the verifier. We use the notation $PK\{(\vec{x}) : F(\vec{x}, \vec{y})\}$ to denote an NIZK for statement F, which reveals $\vec{y}$ but not $\vec{x}$.

A.9. Proofs of Shuffle Correctness

Zero-knowledge proofs of shuffle correctness (also sometimes known as mix proofs) were first introduced by Chaum [49] in the context of mix networks. More formally, let $C_1, \ldots, C_n$ be a sequence of ciphertexts and $C'_1, \ldots, C'_n$ be a permuted and rerandomized version of thereof. Let furthermore π be the used permutation and $r_1, \ldots, r_n$ be the randomnesses used in the rerandomization. A zero-knowledge proof of shuffle correctness can be expressed via NIZKs as follows:

$$PK\left\{(\pi, r_1, \ldots, r_n) : \forall 1 \leq i \leq n.\ C'_i = \mathsf{Rnd}(pk, C_{\pi^{-1}(i)}, r_i)\right\}.$$

Notice that this proof reveals the old and the new ciphertext but it hides π and $r_1, \ldots, r_n$.

A.10. Private Information Retrieval

A private information retrieval (PIR) protocol [53] is defined as follows.

Definition A.17 (Private information retrieval)**.** *A* private information retrieval *protocol consists of the following* PPT *algorithms* $\Pi_{\mathsf{PIR}} = (\mathsf{prepRead}, \mathsf{execRead}, \mathsf{decodeResp})$ *where the query preparation algorithm* $(q, td) \leftarrow \mathsf{prepRead}(\mathcal{DB}, i)$ *on input a database* $\mathcal{DB}$ *(the length of* $\mathcal{DB}$ *usually suffices) and an index* i *outputs a query* q *and a trapdoor* td*; the query execution algorithm* $r \leftarrow \mathsf{execRead}(\mathcal{DB}, q)$ *on input the actual database* $\mathcal{DB}$ *and a query* q *outputs an encoded response* r*; and the response decoding algorithm* $d \leftarrow \mathsf{decodeResp}(td, r)$ *on input the trapdoor* td *and an encoded response* r *outputs a decoded response* d*.*

A PIR is correct if for all indices i, databases $\mathcal{DB}$, $(q, td) \leftarrow \mathsf{prepRead}(\mathcal{DB}, i)$, $r \leftarrow \mathsf{execRead}(\mathcal{DB}, q)$, and $d \leftarrow \mathsf{decodeResp}(td, r)$ we have that $d = \mathcal{DB}(i)$.

Usually it is clear from the context that the query generator and response decoder is the same party, in which case we omit td for brevity.

Security of PIR is defined by requiring that any two queries are indistinguishable to the one executing them.

Definition A.18 (Privacy)**.** *Let* $\Pi_{\mathsf{PIR}} = (\mathsf{prepRead}, \mathsf{execRead}, \mathsf{decodeResp})$ *be a PIR scheme.* Π_{PIR} *achieves* privacy *if for any* PPT *adversary* $\mathcal{A}$ *the following probability is negligible in* λ

$$\left|\Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{PIR}}}_{\mathcal{A},\mathsf{priv}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\Pi_{\mathsf{PIR}}}_{\mathcal{A},\mathsf{priv}}(\lambda, 1) = 1\right]\right|$$

where $\mathsf{Exp}^{\Pi_{\mathsf{PIR}}}_{\mathcal{A},\mathsf{priv}}(\lambda, b)$ *is the following experiment:*

```
$\mathsf{Exp}^{\Pi_{\mathsf{PIR}}}_{\mathcal{A},\mathsf{priv}}(\lambda, b)$
$(\mathcal{DB}, i_0, i_1) \leftarrow \mathcal{A}$
$(q^*, td^*) \leftarrow \mathsf{prepRead}(\mathcal{DB}, i_b)$
$b' \leftarrow \mathcal{A}(q^*)$
if $|\mathcal{DB}| = n \wedge 1 \leq i_0 \leq n \wedge 1 \leq i_1 \leq n \wedge b = b'$ then
  return $1$
return $0$
```

B. Further Details of Group ORAM

B.1. Cryptographic Instantiations

B.1.1. Instantiations

Encryption schemes. We use AES [63] as private-key encryption scheme with an appropriate message padding in order to achieve the elusive-range property [134].[1] Furthermore, we employ the ElGamal encryption scheme [76] for public-key encryption. We use it for PIR-GORAM to construct an entry in the database (cf. $\mathbf{c}_{\mathsf{Data}}$ in (2.3) and $\mathbf{c}_{\mathsf{BrCast}}$ in (2.4)). It also fulfills all properties that we require for GORAM, i.e., it is rerandomizable and supports zero-knowledge proofs. A complete review of the scheme is reported in Figure C.1 on page 181. Interestingly enough, it is also possible to produce information with which one can decrypt a ciphertext $c = (c_1, c_2)$ without knowing the secret key by sending c_1^{-x} where x is the decryption key. This is necessary to give the server access to c_{Auth} in GORAM. In PIR-GORAM, we encrypt the signing keys of the Schnorr signature scheme [173] (cf. $\mathbf{c}_{\mathsf{Auth}}$ in (2.5)) using the Cramer-Shoup encryption scheme [60].

For GORAM and A-GORAM, we utilize the predicate encryption scheme introduced by Katz *et al.* [113]. Its ciphertexts are rerandomizable and we also show them to be compatible with the Groth-Sahai proof system [94]. For the details, we refer to Appendix B.1. Concerning the implementation, the predicate encryption scheme by Katz *et al.* [113] is not efficient enough since it relies on elliptic curves on composite-order groups. In order to reach a high security parameter, the composite-order setting requires us to use much larger group sizes than in the prime-order setting, rendering the advantages of elliptic curves practically useless. Therefore, we use a scheme transformation proposed by David Freeman [82], which works in prime-order groups and is more efficient. For implementing S-GORAM we use an adaptively secure broadcast encryption scheme by Gentry and Waters [85].

Private information retrieval. We use XPIR [2], the state of the art in computational PIR.

Zero-knowledge proofs. We deploy several non-interactive zero-knowledge proofs. For PIR-GORAM, in order to implement the integrity proofs in (cf. lines 4.20 and 4.26 in Section 2.4), we use an OR-proof [58] over a conjunction of plaintext-equivalence proofs [108] (PEP) on the ElGamal ciphertexts forming one entry and a standard discrete logarithm proof [173] showing that the client knows the signing key corresponding to the authenticated verification

[1]This property is formally necessary when proving the hybrid version of our constructions tamper-resistant. We refer to [140, Proof of Lemma 3] for the full proof in the hybrid version.

key. When improving the proof computation using our new technique based on the hash-and-proof paradigm,[2] the conjunction of PEPs reduces to the computation of the homomorphic hash plus one PEP. As a matter of fact, since the public components necessary to verify a proof (the new and old ciphertexts and the verification key) and the secret components necessary to compute the proof (the randomness used for rerandomization or the signing key) are independent of the number of clients, all deployed proofs solely depend on the block size.

In GORAM, for proving that a predicate ciphertext validly decrypts to 1 without revealing the key, we use Groth-Sahai non-interactive zero-knowledge proofs[3] [94]. More precisely, we apply them in the proofs created in line 10.14 (read and write, see Algorithm 10 and Algorithm 11). We employ plaintext-equivalence proofs (PEPs) [108, 173] for the proofs in line 10.16. Furthermore, we use a proof of shuffle correctness [22], batched shuffle proofs, and the hash-and-proof paradigm in lines 10.11 and 11.10.

Chameleon signatures. We use a chameleon hash function by Nyberg and Rueppel [13], which has the key-exposure freeness property. We complete the chameleon hash tags with SHA-256 for the ordinary hash function and combine both with RSA signatures [165].

Implementing permanent entries in GORAM. We briefly outline how permanent entries can be implemented using ElGamal encryption and equality of discrete logarithm proofs [59]. Let $c_p = \mathsf{E}(pk, \mathsf{permanent}) = (G, H) = (g^r, g^{\mathsf{permanent}} \cdot h^r)$ be the ciphertext associated to the entry that is subject to change and $pk = (g, h)$ be the public key of the ElGamal scheme. If $\mathsf{permanent} \neq 1$ then the entry may not be removed from the database completely. Hence, if $\mathcal{O}$ attempts to remove an entry from the tree, she has to prove to $\mathcal{S}$ that $\mathsf{permanent} = 1$. The following zero-knowledge proof serves this purpose, given that permanent is encoded in the exponent of the message:

$$PK\left\{(\alpha) :\ H \cdot g^{-1} = G^{\alpha}\ \wedge\ h = g^{\alpha}\right\}.$$

Naturally, the re-randomization step as well as the shuffle proof step also apply to this ciphertext.

B.1.2. Details on the KSW Predicate Encryption Scheme

In this section, we present the predicate encryption scheme of Katz *et al.* [113]. Moreover, we show how one can rerandomize ciphertexts as a public operation. This might be of independent interest. We show the rerandomization for the original scheme, however, it is straightforward to adapt the transformation to prime order groups [82] – the one we use in our implementation for efficiency reasons.

Importantly, in our descriptions throughout the paper, we use a zero-knowledge proof of knowledge of a secret key for the predicate encryption scheme, showing that the result of

[2]A careful analysis of the computation shows that the technique from batched shuffle proofs mapped to standard PEPs as we deploy them in PIR-GORAM, would end up in a worse solution.

[3]Groth-Sahai proofs are generally not zero-knowledge. However, in our case the witnesses fulfill a special equation for which they are zero-knowledge.

decryption is the neutral element. Unfortunately, straightforwardly combining the Groth-Sahai proof system [94], which we deploy in the concrete instantiation, with the standard predicate encryption scheme of Katz *et al.* [113] only yields witness-indistinguishability proofs since the right hand side of the underlying pairing product equation is not the neutral element. Only if it was the neutral element, the proof would be zero-knowledge. Fortunately, Katz *et al.* also present a version of the predicate encryption scheme that does not encrypt messages but just the attribute, which can be seen as the encryption of the neutral element. This scheme combined with Groth-Sahai proofs yields zero-knowledge proofs of knowledge. We thus present both schemes in the sequel.

B.1.2.1. The Scheme Instantiation

The scheme is based on composite order groups with a bilinear map. More precisely, let $N = pqr$ be a composite number where p, q, and r are large prime numbers. Let $\mathbb{G}$ be an order-N cyclic group and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Recall that e is *bilinear*, i.e., $e(g^a, g^b) = e(g, g)^{ab}$, and *non-degenerate*, i.e., if $\langle g \rangle = \mathbb{G}$ then $e(g, g) \neq 1$. Then, by the chinese remainder theorem, $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$ where $\mathbb{G}_s$ with $s \in \{p, q, r\}$ are the s-order subgroups of $\mathbb{G}$. Moreover, given a generator g for $\mathbb{G}$, $\langle g^{pq} \rangle = \mathbb{G}_r$, $\langle g^{pr} \rangle = \mathbb{G}_q$, and $\langle g^{qr} \rangle = \mathbb{G}_p$. Another insight is the following, given for instance $a \in \mathbb{G}_p$ and $b \in \mathbb{G}_q$, we have $e(a, b) = e((g^{qr})^c, (g^{pr})^d) = e(g^{rc}, g^d)^{pqr} = 1$, i.e., a pairing of elements from different subgroups cancels out. Finally, let $\mathcal{G}$ be an algorithm that takes as input a security parameter 1^λ and outputs a description $(p, q, r, \mathbb{G}, \mathbb{G}_T, e)$. We describe the algorithms $\mathsf{Gen}_{\mathsf{PE}}$, K_{PE}, E_{PE}, and D_{PE} in the sequel.

Algorithm $\mathsf{Gen}_{\mathsf{POE}}(1^\lambda, n)$ **and** $\mathsf{Gen}_{\mathsf{PE}}(1^\lambda, n)$**.** First, the algorithm runs $\mathcal{G}(1^\lambda)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, e)$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Then, it computes g_p, g_q, and g_r as generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively. The algorithm selects $R_0 \in \mathbb{G}_r$, $R_{1.i}, R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}, h_{2,i} \in \mathbb{G}_p$ uniformly at random for $1 \leq i \leq n$. $(N = pqr, \mathbb{G}, \mathbb{G}_T, e)$ constitutes the public parameters. The public key for the predicate-only encryption scheme is

$$opk = (g_p, g_r, Q = g_q \cdot R_0, \{H_{1,i} = h_{1,i} \cdot R_{1,i}, H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1}^n)$$

and the master secret key is

$$omsk = (p, q, r, g_q, \{h_{1,i}, h_{2,i}\}_{i=1}^n).$$

For the predicate encryption with messages, the algorithm additionally chooses $\gamma \in \mathbb{Z}_N$ and $h \in \mathbb{G}_p$ at random. The public key is

$$ppk = (g_p, g_r, Q = g_q \cdot R_0, P = e(g_p, h)^\gamma, \{H_{1,i} = h_{1,i} \cdot R_{1,i}, H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1}^n)$$

and the master secret key is

$$pmsk = (p, q, r, g_q, h^{-\gamma}, \{h_{1,i}, h_{2,i}\}_{i=1}^n).$$

Algorithm $\mathsf{K}_{\mathsf{POE}}(omsk, \vec{v})$ **and** $\mathsf{K}_{\mathsf{PE}}(pmsk, \vec{v})$**.** Parse $\vec{v}$ as $(v_1, \ldots, v_n)$ where $v_i \in \mathbb{Z}_N$. The algorithm picks random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $1 \leq i \leq n$, random $R_5 \in \mathbb{G}_r$, random $f_1, f_2 \in \mathbb{Z}_q$, and random $Q_6 \in \mathbb{G}_q$. For the predicate-only encryption scheme it outputs a secret key

$$osk_{\vec{v}} = \left(K_0 = R_5 \cdot Q_6 \cdot \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \{K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot v_i}, K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot v_i}\}_{i=1}^{n} \right).$$

For the predicate encryption scheme with messages, the secret key $psk_{\vec{v}}$ is the same as $osk_{\vec{v}}$ except for

$$K_0 = R_5 \cdot Q_6 \cdot h^{-\gamma} \cdot \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}.$$

Algorithm $\mathsf{E}_{\mathsf{POE}}(opk, \vec{x})$ **and** $\mathsf{E}_{\mathsf{PE}}(ppk, \vec{x}, m)$**.** Parse $\vec{x}$ as $(x_1, \ldots, x_n)$ where $x_i \in \mathbb{Z}_N$. The algorithm picks random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $1 \leq i \leq n$. For the predicate-only encryption scheme it outputs the ciphertext

$$C = \left(C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i}\}_{i=0}^{n} \right).$$

For the predicate encryption scheme with messages notice that $m \in \mathbb{G}_T$. The ciphertext is

$$C = \left(C' = m \cdot P^s, C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i}\}_{i=0}^{n} \right).$$

Algorithm $\mathsf{D}_{\mathsf{POE}}(osk_{\vec{v}}, C)$ **and** $\mathsf{D}_{\mathsf{PE}}(psk_{\vec{v}}, C)$**.** The predicate-only encryption outputs whether the following equation is equal to 1

$$e(C_0, K_0) \cdot \prod_{i=1}^{n} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}).$$

The predicate encryption scheme with messages outputs the result of the following equation

$$C' \cdot e(C_0, K_0) \cdot \prod_{i=1}^{n} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}).$$

Correctness. We show correctness for both schemes in one calculation, indicating additional elements that are present in predicate encryption with messages using boxes. We have secret key components $(K_0, \{K_{1,i}, K_{2,i}\}_{i=0}^{n})$ where $K_0 = R_5 \cdot Q_6 \cdot \boxed{h^{-\gamma}} \cdot \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}$ and ciphertext components $(\boxed{C'}, C_0, \{C_{1,i}, C_{2,i}\}_{i=0}^{n})$. Then

$$\begin{aligned}
& \boxed{C'} \cdot e(C_0, K_0) \cdot \prod_{i=0}^{n} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}) \\
= \; & \boxed{m \cdot P^s} \cdot e(g_p^s, R_5 \cdot Q_6 \cdot \boxed{h^{-\gamma}} \cdot \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}) \cdot \prod_{i=1}^{n} e(H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot v_i}) \\
& \cdot e(H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i}, g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot v_i}) \\
= \; & \boxed{m \cdot e(g_p, h)^{s \cdot \gamma} \cdot e(g_p, h)^{-s \cdot \gamma}} \cdot \prod_{i=1}^{n} e(g_p, h_{1,i})^{-s \cdot r_{1,i}} \cdot e(g_p, h_{2,i})^{-s \cdot r_{2,i}}
\end{aligned}$$

$$\begin{aligned}
&\cdot \prod_{i=1}^{n} e(h_{1,i}, g_p)^{s\cdot r_{1,i}} \cdot e(g_q, g_q)^{\alpha\cdot x_i\cdot f_1\cdot v_i} \cdot e(h_{2,i}, g_p)^{s\cdot r_{2,i}} \cdot e(g_q, g_q)^{\beta\cdot x_i\cdot f_2\cdot v_i} \\
= \quad & \boxed{m} \cdot \prod_{i=1}^{n} e(g_q, g_q)^{(\alpha f_1+\beta f_2)\cdot x_i\cdot v_i} \\
= \quad & \boxed{m} \cdot e(g_q, g_q)^{(\alpha f_1+\beta f_2)\cdot\langle\vec{x},\vec{v}\rangle}
\end{aligned}$$

The second factor in the last line cancels out only if $\langle\vec{x},\vec{v}\rangle = 0$, as expected.

B.1.2.2. Rerandomizing KSW Ciphertexts

The correctness validation in the previous section already suggests that rerandomization of Π_{POE} and Π_{PE} ciphertexts is possible since all terms that involve randomness s cancel out in the end. As terms including s only occur in the ciphertext we can easily have public rerandomization functions $\mathsf{R}_{\mathsf{POE}}$ and R_{PE} as follows.

Algorithms $\mathsf{R}_{\mathsf{POE}}(C)$ and $\mathsf{R}_{\mathsf{PE}}(C)$. The algorithm picks fresh randomness $s' \in \mathbb{Z}_N$ and computes C_R in the predicate-only encryption scheme as

$$C_R = (C_0 \cdot g_p^{s'}, \{C_{1,i} \cdot H_{1,i}^{s'}, C_{2,i} \cdot H_{2,i}^{s'}\}_{i=1}^{n}).$$

In the predicate encryption scheme with messages it returns

$$C_R = (C' \cdot P^{s'}, C_0 \cdot g_p^{s'}, \{C_{1,i} \cdot H_{1,i}^{s'}, C_{2,i} \cdot H_{2,i}^{s'}\}_{i=1}^{n}).$$

This transformation constitutes an additive randomization in the sense that in every exponent where s occurs, it now contains exponent $s + s'$. Therefore, also the correctness is preserved.

B.1.2.3. Proving Knowledge of Predicate-Only Secret Keys in Groth-Sahai

In our construction, the client has to prove to the server that she is eligible to write an entry whenever she wants to replace an entry in the database. The proof (see line 10.14) has the general form

$$PK\left\{\left(psk_f\right) :\ \mathsf{D}_{\mathsf{PE}}(psk_f, c_{\mathsf{Auth}}) = 1\right\}.$$

In our instantiation we have to use KSW predicate-only encryption for the reasons mentioned in the beginning, hence the statement changes to

$$PK\left\{\left(osk_f\right) :\ \mathsf{D}_{\mathsf{POE}}(osk_f, c_{\mathsf{Auth}}) = \top\right\}.$$

Concretely, c_{Auth} is of the form $(C_0, \{C_{1,i}\}_{1\le i\le n}, \{C_{2,i}\}_{1\le i\le n})$ while secret keys are of the form $(K_0, \{K_{1,i}\}_{1\le i\le n}, \{K_{2,i}\}_{1\le i\le n})$. This means that the concrete proof is of the form

$$PK\left\{(K_0, \{K_{1,i}\}_{1\le i\le n}, \{K_{2,i}\}_{1\le i\le n}) :\ e(C_0, K_0) \cdot \prod_{i=1}^{n} e(C_{1,i}, K_{1,i}) e(C_{2,i}, K_{2,i}) = 1\right\}.$$

The Groth-Sahai proof system [94] allows for proving relations of the above form. More precisely, given vectors of witnesses $\vec{X} \in \mathbb{G}_1^m$, $\vec{Y} \in \mathbb{G}_2^n$, we can prove the following equality while disclosing neither $\vec{X}$ nor $\vec{Y}$:

$$\prod_{i=1}^{n} e(A_i, Y_i) \cdot \prod_{i=1}^{m} e(X_i, B_i) \cdot \prod_{i=1}^{n} \prod_{j=1}^{m} e(X_i, Y_j)^{\gamma_{ij}} = t_T$$

where $\vec{A} \in \mathbb{G}_1^n$, $\vec{B} \in \mathbb{G}_2^m$, $\Gamma \in \mathbb{Z}_q^{n \times m}$, and $t_T \in \mathbb{G}_T$ are the public components of the proof. In our special case, it is sufficient to consider the following special form of this equation since we only want to keep the secret key hidden, which is in $\mathbb{G}_2$:

$$\prod_{i=1}^{n} e(A_i, Y_i) = t_T.$$

Furthermore, $t_T = 1$ where 1 stands for the neutral element of the group operation. We construct the vectors $\vec{A}$ and $\vec{Y}$ as

$$\vec{A} = (C_0, C_{1,i}, \ldots, C_{1,n}, C_{2,1}, \ldots, C_{2,n})^\top \qquad \vec{Y} = (K_0, K_{1,i}, \ldots, K_{1,n}, K_{2,1}, \ldots, K_{2,n})^\top$$

We do not review the proof construction here but refer the interested reader to [94] for a concise explanation. We observe that since $t_T = 1$, the proofs for our scenario are indeed zero-knowledge.

B.2. Proof of the Computational Lower Bound

Notation. Without loss of generality we assume a binary database $\mathcal{DB}$ that consists of n entries and each entry consists of exactly one bit d. We denote by $\mathcal{I}$ the set of identifiers j of each entry and by d_j the value of j. Note that a single entry can be simultaneously stored on multiple physical addresses in the memory of the database (e.g., the database may maintain multiple copies of the same entry or secret-share some entries across several locations), for this reason we define a predicate that maps identifiers to physical addresses of the memory of the database. Let $\mathcal{L}$ denote the set of possible physical memory locations. The predicate $\mathsf{loc} : \mathcal{I} \to \mathcal{P}(\mathcal{L})$ takes as input an identifier j and returns a set of physical memory addresses $\{\ell_1, \ldots, \ell_t\}$, for some $t \geq 1$. Note that such a predicate may depend on the database architecture and on the ORAM scheme and it may change whenever some *write* operation is performed on the database. We say that an algorithm *accesses* some physical address if its content is either modified or read during the execution of the algorithm. Throughout the following presentation, we denote by L a given set of physical addresses and by m the amount of physical addresses of the database, i.e., $m = |\mathcal{L}|$. We observe that, by definition, we have that $m \geq n$. Finally, for a positive integer n, we let $[n]$ denote the set $\{1, \ldots, n\}$. We also extend this notation to sets of indexed variables; we write, e.g., $[cap_k]$ to denote the set $\{cap_1, \ldots, cap_k\}$.

Read Correctness. We recall the correctness definition for the read operation in a Group ORAM, while for the other operations we refer to Appendix B.4.1.

Definition B.1 (Read Correctness)**.** *The read of a Group ORAM scheme* Π_{GORAM} *is* correct*, if for all* λ *and* n*, for all* $j \in [n]$ *and* $cap_i \in [cap_k]$ *such that* $\mathbf{AC}(i,j) \neq \bot$*, there exists a negligible function* $\mu(\cdot)$ *such that:*

$$\mathsf{Pr}\left[d_j \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle\right] \geq (1 - \mu(\lambda)).$$

We now introduce some helpful facts which simplify the proof of the main theorem. Note that in the following analysis we only consider the operation read, since read and write must be indistinguishable it is easy to see that the same result extends to write. The first lemma formalizes that in any Group ORAM scheme, a client who wants to read a certain index necessarily has to access at least one of the physical addresses associated to that index.

Lemma B.1. *Let* Π_{GORAM} *be a Group ORAM scheme. Then for all* $j \in [n]$ *and* $cap_i \in [cap_k]$ *with* $\mathbf{AC}(i,j) \neq \bot$ *there exists a negligible function* $\mu(\cdot)$ *such that:*

$$\mathsf{Pr}\left[d_j \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle \text{ accesses } L \wedge L \cap \mathsf{loc}(j) \neq \emptyset\right] \geq (1 - \mu(\lambda))$$

where the probability is taken over the random coins of $\langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$*.*

Proof. The proof follows from the correctness of Π_{GORAM}. Assume towards contradiction that $\langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ does not access any of the physical addresses in $\mathsf{loc}(j)$ with non-negligible probability $\epsilon(\lambda)$. In that case, the algorithm $\langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ can only guess the bit of j, and hence returns the correct bit with probability at most $1/2$, which means that $\langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ returns d_j with probability at most $(1-\epsilon(\lambda)/2)$. Since $\epsilon(\lambda)/2$ is non-negligible, we derived a contradiction to Π_{GORAM}'s read correctness. □

The following lemma captures the fact that in a Group ORAM the probability of a random physical address to be accessed during a certain read is proportional to the amount of entries accessed (on average) upon each read.

Lemma B.2. *Let* Π_{GORAM} *be a Group ORAM scheme whose* read *operation accesses on average* ℓ *many addresses (for some* $\ell \in [m]$*), over the random coins of the* read *operation. Then for a physical memory address* $x \in [m]$ *sampled uniformly at random and for all* read *access sequences* $\vec{y}$ *of length* p*, for all* $q \in [p]$ *it holds that:*

$$\mathsf{Pr}\left[\vec{y}_q \text{ accesses } x\right] \leq \frac{\ell}{m}$$

where $\vec{y}_q$ *denotes the* q*-th* read *operation of* $\vec{y}$ *and the probability is taken over the random choice of* x *and the random coins of* $\vec{y}$*.*

Proof. The proof follows from the fact that x is sampled independently and uniformly at random. Since on average $\langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ accesses ℓ many memory locations, the probability that x belongs to that set is exactly ℓ/m. □

The following lemma formalizes the intuition that in a Group ORAM scheme that is oblivious (against malicious clients) the probability of accessing a certain memory address upon each read must be independent from the index queried by the client.

Lemma B.3. *Let Π_{GORAM} be a Group ORAM scheme, then for all pairs of* read *access sequences $(\vec{y}, \vec{z})$ of length p, for all $q \in [p]$, and for all memory locations $x \in [m]$ there exists a negligible function $\mu(\cdot)$ such that:*

$$|\mathsf{Pr}[\vec{y}_q \textit{ accesses } x] - \mathsf{Pr}[\vec{z}_q \textit{ accesses } x]| \leq \mu(\lambda)$$

where the probability is taken over the random coins of $\langle \mathcal{C}_{\mathsf{read}}(\cdot,\cdot), \mathcal{S}(\mathcal{DB})\rangle$.

Proof. Assume towards contradiction that there exists a pair of access sequences $(\vec{y}, \vec{z})$ and a physical memory address x such that at step q:

$$|\mathsf{Pr}[\vec{y}_q \textit{ accesses } x] - \mathsf{Pr}[\vec{z}_q \textit{ accesses } x]| \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\cdot)$. This implies that we can construct an adversary $\mathcal{A}$ that non-deterministically samples $(\vec{y}, \vec{z}, x, q)$, does not corrupt any client, queries the pair $(\vec{y}, \vec{z})$ to the query interface of the challenger and returns 1 if x was accessed at step q and 0 otherwise. To analyze the success probability of $\mathcal{A}$ we shall note that:

$$\mathsf{Pr}[1 \leftarrow \mathcal{A} \mid b = 0] = \mathsf{Pr}[\vec{y}_q \textit{ accesses } x]$$
$$\mathsf{Pr}[1 \leftarrow \mathcal{A} \mid b = 1] = \mathsf{Pr}[\vec{z}_q \textit{ accesses } x].$$

By our initial assumption it follows that:

$$|\mathsf{Pr}[1 \leftarrow \mathcal{A} \mid b = 0] - \mathsf{Pr}[1 \leftarrow \mathcal{A} \mid b = 1]| \geq \epsilon(\lambda),$$

which is a contradiction to the *obliviousness against malicious clients* of Π_{GORAM}. □

We are now ready to formally prove Theorem 2.1. Note that, even though the definition of obliviousness against malicious clients for Group ORAM (see Definition 2.6) allows for active corruption, the result still holds in case of passive corruption (i.e., the adversary does not impersonate the corrupted instances but receives transcripts of honestly executed operations).

Proof. Assume towards contradiction that there exists a Group ORAM scheme Π_{GORAM} that accesses on average $o(n)$ physical memory addresses and is secure. Then we can construct an adversary $\mathcal{A}$ as follows.

$\mathcal{A}$ receives a local copy of the database $\mathcal{DB}$. $\mathcal{A}$ then generates several clients via the addCl interface and samples a pair of entries $(j, j') \leftarrow [n]^2$ uniformly at random. It then selects a random client with identifier i and assigns i with read permissions on j via the interface chMode. Furthermore, $\mathcal{A}$ samples a memory location x uniformly at random, an integer $q \in \{1, \ldots, \text{poly}(\lambda)\}$, and a sequence $\vec{w}$ of read operations uniformly at random of length $q-1$. $\mathcal{A}$ then initializes $\vec{y} := \vec{w}||(i', j, \perp)$ and $\vec{z} := \vec{w}||(i', j', \perp)$ for some client identifier $i' \neq i$ such that $\mathbf{AC}(i', j) = \mathbf{AC}(i', j') = \mathsf{R}$. If such an i' does not exists, then $\mathcal{A}$ generates a new non-corrupted client i' with the appropriate read permissions via the interfaces addCl and chMode. $\mathcal{A}$ queries $(\vec{y}, \vec{z})$ to the query interface. After the $(q-1)$-th step, the adversary corrupts client i and locally simulates read(j, cap_i): if x is not accessed, then $\mathcal{A}$ interrupts the simulation and outputs a random guess. Otherwise $\mathcal{A}$ observes the blocks accessed during the execution of the q-th operation: if x is accessed $\mathcal{A}$ returns 1, otherwise he returns 0.

For the analysis, it is easy to see that $\mathcal{A}$ is efficient. To analyze the success probability of $\mathcal{A}$ we define GUESS to be the event when the sampled block x belongs to the physical locations of the entry j and it is accessed in both the simulated read and $\vec{y}$. More formally, GUESS is the event where $x \in \mathsf{loc}(j)$ and $x \in L$ and $x \in L'$, where L and L' are the set of physical addresses accessed by $\mathsf{read}(cap_i, j)$ and $\mathsf{read}(cap_{i'}, j)$ respectively. Then we can express the probability that $\mathcal{A}$ outputs 1 given that the challenger is executing $\vec{y}$ as:

$$\begin{aligned}\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 0\right] = {} & \mathsf{Pr}\left[\vec{y} \textit{ accesses } x \mid \mathsf{GUESS}\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] \\ & + \mathsf{Pr}\left[\vec{y} \textit{ accesses } x \mid \overline{\mathsf{GUESS}}\right] \mathsf{Pr}\left[\overline{\mathsf{GUESS}}\right].\end{aligned}$$

By definition of GUESS we have

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 0\right] = 1 \cdot \mathsf{Pr}\left[\mathsf{GUESS}\right] + \mathsf{Pr}\left[\vec{y} \textit{ accesses } x \mid \overline{\mathsf{GUESS}}\right] \mathsf{Pr}\left[\overline{\mathsf{GUESS}}\right]. \tag{B.1}$$

Now we express the probability of the adversary to output 1 given that the challenger is executing $\vec{z}$:

$$\begin{aligned}\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right] = {} & \mathsf{Pr}\left[\vec{z} \textit{ accesses } x \mid \mathsf{GUESS}\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] \\ & + \mathsf{Pr}\left[\vec{z} \textit{ accesses } x \mid \overline{\mathsf{GUESS}}\right] \mathsf{Pr}\left[\overline{\mathsf{GUESS}}\right].\end{aligned}$$

We shall note that the memory block x is uniformly distributed over the memory of the database, in particular it holds that x and j' are independently and uniformly sampled. Thus by Lemma B.2 we can rewrite:

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right] \leq \frac{\ell}{m} \cdot \mathsf{Pr}\left[\mathsf{GUESS}\right] + \mathsf{Pr}\left[\vec{z} \textit{ accesses } x \mid \overline{\mathsf{GUESS}}\right] \mathsf{Pr}\left[\overline{\mathsf{GUESS}}\right].$$

Since we assumed Π_{GORAM} to be oblivious by Lemma B.3 we have

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right] \lesssim \frac{\ell}{m} \cdot \mathsf{Pr}\left[\mathsf{GUESS}\right] + \mathsf{Pr}\left[\vec{y} \textit{ accesses } x \mid \overline{\mathsf{GUESS}}\right] \mathsf{Pr}\left[\overline{\mathsf{GUESS}}\right] \tag{B.2}$$

where $\lesssim$ is like $\leq$ but neglects additive negligible terms. If we consider the difference of the two probabilities, Equations (B.1) and (B.2) cancel each other out and we are left with:

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right]\right| \gtrsim \left|\mathsf{Pr}\left[\mathsf{GUESS}\right] - \frac{\ell}{m}\mathsf{Pr}\left[\mathsf{GUESS}\right]\right|.$$

Since by definition the two terms are both non-negative we have:

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right]\right| \gtrsim \mathsf{Pr}\left[\mathsf{GUESS}\right]\left(1 - \frac{\ell}{m}\right). \tag{B.3}$$

We now observe that

$$\mathsf{Pr}\left[\mathsf{GUESS}\right] = \mathsf{Pr}\left[x \in L \wedge x \in L' \mid x \in \mathsf{loc}(j)\right] \mathsf{Pr}\left[x \in \mathsf{loc}(j)\right]$$

Since the local simulation of $\mathcal{A}$ and the simulation of the challenger have independent random coins, we can rewrite

$$\mathsf{Pr}\left[\mathsf{GUESS}\right] = \left(\mathsf{Pr}\left[x \in L \mid x \in \mathsf{loc}(j)\right] \cdot \mathsf{Pr}\left[x \in L' \mid x \in \mathsf{loc}(j)\right]\right) \cdot \mathsf{Pr}\left[x \in \mathsf{loc}(j)\right]^2$$

For all logical indices j, we have that the probability that $x \in \mathsf{loc}(j)$ is $\frac{|\mathsf{loc}(j)|}{m}$, thus

$$\mathsf{Pr}\left[\mathsf{GUESS}\right] = \left(\mathsf{Pr}\left[x \in L \mid x \in \mathsf{loc}(j)\right] \cdot \mathsf{Pr}\left[x \in L' \mid x \in \mathsf{loc}(j)\right]\right) \cdot \left(\frac{|\mathsf{loc}(j)|}{m}\right)^2$$

By Lemma B.1 we have that $\mathsf{Pr}\left[x \in L \mid x \in \mathsf{loc}(j)\right] \geq \frac{(1-\mu(\lambda))}{|\mathsf{loc}(j)|}$ over the random choice of x (the same holds for $\mathsf{Pr}\left[x \in L' \mid x \in \mathsf{loc}(j)\right]$). Therefore

$$\mathsf{Pr}\left[\mathsf{GUESS}\right] \geq \left(\frac{(1-\mu(\lambda))}{|\mathsf{loc}(j)|}\right)^2 \left(\frac{|\mathsf{loc}(j)|}{m}\right)^2 = \frac{(1-\mu(\lambda))^2}{m^2}$$

We can now substitute to equation (B.3)

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right]\right| \gtrsim \frac{(1-\mu(\lambda))^2}{m^2} \cdot \left(1 - \frac{\ell}{m}\right) \gtrsim \frac{1}{m^2} \cdot \left(1 - \frac{\ell}{m}\right).$$

By assumption $\ell = o(n)$, since $m \geq n$ then $\frac{\ell}{m}$ is non-negligibly smaller than 1, since $\frac{1}{m^2}$ is also a non-negligible positive value it follows that the difference of probabilities is non-negligible. This is a contradiction with respect to the obliviousness against malicious clients of Π_{GORAM} and it concludes our proof. $\square$

B.3. Security Proofs for Batched Shuffle Proofs and the Hash-and-Proof Paradigm

Proof of Theorem 2.2. Correctness. For our BZKPS, it is easy to see that whenever the two input matrices of ciphers A and B are honestly computed and the two parties do not deviate from the protocol specifications, the probability that the verifier is convinced of the statement does not vary with respect to the underlying ZKP. This follows directly from the fact that, as long as A differs from B only in the order of the rows, the resulting vectors $\vec{a}$ and $\vec{b}$ will contain the same entries, in the same permuted order. Thus correctness holds true.
Zero-Knowledge and proof of knowledge. The zero-knowledge and the proof of knowledge properties can be also derived by the ZKP since the only procedure performed in BZKPS, outside of the original protocol, is a public operation, namely, the multiplication of ciphertexts which causes homomorphically a summation of plaintexts.
Soundness. In order to show that our approach preserves the soundness of the ZKP, we have to prove that any malicious prover trying to fool the protocol succeeds with probability at most $1/2$. It is clear that this fact suffices by itself since with k protocol runs the success probability of the adversary drops to $1/2^k$, which is exponentially low in k. The intuition

behind the proof is that the adversary cannot modify a single block in a row of B without the verifier noticing at least half of the times, thus he needs to modify at least one more spot so as to cancel out the added values in the total sum. But, again, this block will be selected by the verifier to contribute to the sum just half of the times in expectation, so the success probability does not increase. This holds true no matter how many blocks in the same row the adversary modifies.

In the soundness scenario a malicious prover $\mathcal{P}^*_{\mathcal{A}}$ wants to convince the verifier $\mathcal{V}^*$ that the matrices of ciphertexts A and B contain the same plaintexts up to the order of the rows, i.e., they are permuted with respect to π, and the ciphertexts are rerandomized with respect to R. Let $|A| = |B| = n \times m$. Note that B is generated by $\mathcal{P}^*_{\mathcal{A}}$ and then sent to $\mathcal{V}^*$ in an early stage of the protocol. Since the underlying ZKP is assumed to be sound, it follows that, any time an element a_i in the resulting vector $\vec{a}$ differs with respect to the correspondent permuted element $b_{\pi(i)}$ in the vector $\vec{b}$, the verifier will notice it with overwhelming probability. Thus, in order for $\mathcal{P}^*_{\mathcal{A}}$ to succeed, it must be the case that for all i, $[\![a_i]\!] = [\![b_{\pi(i)}]\!]$ where $[\![c]\!]$ denotes the plaintext encrypted in c.

Let Δ_i be the difference between a_i and $b_{\pi(i)}$ and $\delta_i = [\![\Delta_i]\!]$. It follows from the argument above and from the homomorphic property of the encryption scheme that, in order for the protocol to successfully terminate, for all i, δ_i must be equal to 0. Since $\mathcal{P}^*_{\mathcal{A}}$ has no control over the input A, and therefore over $\vec{a}$, the value of δ_i is directly derived from the modifications that $\mathcal{P}^*_{\mathcal{A}}$ introduced in the i-th row of the matrix B. We next prove the following: if $\mathcal{P}^*_{\mathcal{A}}$ performed at least one modification on a given row i of B (i.e., he added some value different from zero to the plaintext of $B_{i,j}$ for some $1 \leq j \leq m$), then δ_i has any possible fixed value with probability at most 1/2. Intuitively, this statement is true since with probability 1/2, the verifier does not pick a column that contributes to the difference of the plaintext sum. Since any δ_i must be zero for $\mathcal{P}^*_{\mathcal{A}}$ to succeed, this directly proves our theorem.

The proof is conducted by induction on ℓ, the number of modified blocks in the given row:

$\ell = 1$: Assume without loss of generality that $\mathcal{P}^*_{\mathcal{A}}$ introduces an arbitrary modification $z \neq 0$ on a given block (i.e., on the cipher identified by some column index j), then such a block is selected to contribute to the product b_i with probability exactly 1/2. Thus it holds that:

$$\mathsf{Pr}\left[\delta_i^1 = 0\right] = \frac{1}{2} \qquad\qquad \mathsf{Pr}\left[\delta_i^1 = z\right] = \frac{1}{2}$$

$\ell \to \ell + 1$: Assume without loss of generality that $\mathcal{P}^*_{\mathcal{A}}$ introduces some arbitrary modifications different than zero on ℓ blocks of the given row and that he added the value z to the $(\ell + 1)$-th spot. We denote by $\mathsf{CHOOSE}(\ell)$ the event that the ℓ-th block is picked to contribute to the sum b_i. Then, for all values y it holds that

$$\begin{aligned}\mathsf{Pr}\left[\delta_i^{\ell+1} = y\right] = \mathsf{Pr}\left[\delta_i^{\ell} = y - z \mid \mathsf{CHOOSE}(\ell+1)\right] \cdot \mathsf{Pr}\left[\mathsf{CHOOSE}(\ell+1)\right] +\\ \mathsf{Pr}\left[\delta_i^{\ell} = y \mid \neg\, \mathsf{CHOOSE}(\ell+1)\right] \cdot \mathsf{Pr}\left[\neg\, \mathsf{CHOOSE}(\ell+1)\right].\end{aligned}$$

Since $z \neq 0$ it follows that $y \neq y - z$, additionally the $(\ell+1)$-th spot is chosen to contribute to the sum b_i with probability $1/2$, so we have

$$\mathsf{Pr}\left[\delta_i^{\ell+1} = y\right] = \frac{1}{2} \cdot \mathsf{Pr}\left[\delta_i^{\ell} = y - z \mid \mathsf{CHOOSE}(\ell+1)\right] + \frac{1}{2} \cdot \mathsf{Pr}\left[\delta_i^{\ell} = y \mid \neg\ \mathsf{CHOOSE}(\ell+1)\right].$$

Furthermore the two events are independent, thus we can rewrite

$$\mathsf{Pr}\left[\delta_i^{\ell+1} = y\right] = \frac{1}{2} \cdot \mathsf{Pr}\left[\delta_i^{\ell} = y - z\right] + \frac{1}{2} \cdot \mathsf{Pr}\left[\delta_i^{\ell} = y\right].$$

By induction hypothesis it holds that for all i, δ_i has any fixed value with probability at most $\frac{1}{2}$, then

$$\mathsf{Pr}\left[\delta_i^{\ell+1} = y\right] = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}.$$

The induction above proves that for all i, $\delta_i = 0$ with probability at most $1/2$, therefore the verifier notifies the cheating prover with probability at least $1/2$. This concludes our proof. $\square$

Proof of Theorem 2.3. Correctness. The correctness of Π_{PKE} and of the ZKP $(\mathcal{P}, \mathcal{V})$ imply the correctness of the protocol described above.
Zero-knowledge and proof of knowledge. The *zero-knowledge* of the protocol follows from the zero-knowledge of $(\mathcal{P}, \mathcal{V})$.
Soundness. Arguing about the soundness requires a more accurate analysis: we define as $\mathsf{CHEAT}_{(\mathcal{P}^*_{\mathcal{A}}, \mathcal{V}^*)}$ the event where a malicious $\mathcal{P}^*_{\mathcal{A}}$ fools $\mathcal{V}^*$ into accepting a proof over a false statement. This event happens for all $1 \leq i \leq n$ with probability

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{CHEAT}_{(\mathcal{P}^*_{\mathcal{A}}, \mathcal{V}^*)}\right] &= \mathsf{Pr}\left[\mathsf{CHEAT}_{(\mathcal{P}_{\mathcal{A}}, \mathcal{V})} \mid b_i = \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right] \cdot \mathsf{Pr}\left[b_i = \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right] \\ &+ \mathsf{Pr}\left[\mathsf{CHEAT}_{(\mathcal{P}_{\mathcal{A}}, \mathcal{V})} \mid b_i \neq \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right] \cdot \mathsf{Pr}\left[b_i \neq \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right]\end{aligned}$$

where the probabilities are taken over the random coins of $\mathcal{P}^*_{\mathcal{A}}$ and $\mathcal{V}^*$. By the soundness of $(\mathcal{P}, \mathcal{V})$ we get

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{CHEAT}_{(\mathcal{P}^*_{\mathcal{A}}, \mathcal{V}^*)}\right] &\leq 1 \cdot \mathsf{Pr}\left[b_i = \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right] + \mu \cdot \mathsf{Pr}\left[b_i \neq \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right] \\ &\leq \mu + \mathsf{Pr}\left[b_i = \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right]\end{aligned}$$

where μ is a negligible function in the security parameter. Therefore, to prove soundness, it is sufficient to show that when $\mathsf{CHEAT}_{(\mathcal{P}^*_{\mathcal{A}}, \mathcal{V}^*)}$ happens $\mathsf{Pr}\left[b_i = \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right]$ is a negligible function in the security parameter. We shall note that, due to the homomorphic properties of Π_{PKE}, the resulting plaintext of b_i and $a_{\pi^{-1}(i)}$ are $z_0 + \sum_{j=1}^{m} z_{j+1} \mathsf{D}(dk, B_{i,j}) \in \mathbb{F}_p$, and $z_0 + \sum_{j=1}^{m} z_{j+1} \mathsf{D}(dk, A_{\pi^{-1}(i),j}) \in \mathbb{F}_p$, respectively. It is easy to see that this corresponds to the computation of the *universal pairwise hash function* $\mathsf{h}_{(\vec{z})}$ as described by Carter and Wegman in [44] (Proposition 8). It follows that for all rows $B_i \neq A_{\pi^{-1}(i)}$ the resulting plaintexts of b_i and $a_{\pi^{-1}(i)}$ are uniformly distributed over $\mathbb{F}_p$, thus $\mathsf{Pr}\left[b_i = \mathsf{Rnd}(ek, a_{\pi^{-1}(i)}, r_i)\right] = p^{-2}$, which is a negligible function in the security parameter. Since there are only polynomially many rows, this concludes our proof. $\square$

B.4. Security Proofs

In this section we formally define and sketch the proof for the correctness of our scheme (Section B.4.1) and we prove the theorems presented in Section 2.9 (Section B.4.2).

B.4.1. Correctness

In the following we state the conditions that determine the correctness of a multi-client ORAM. Intuitively, the primitive is correct if, given a successful execution of any algorithm, its outcome satisfies some specific constraints which guarantee the reliability of the whole scheme. We formalize the notion of correctness up to each operation defined within the multi-client ORAM.

Definition B.2 (Correctness). *A Group ORAM* $(\mathsf{gen}, \mathsf{addCl}, \mathsf{addE}, \mathsf{chMode}, \mathsf{read}, \mathsf{write})$ *is* correct*, if the following statements are true except with negligible probability in the security parameter: let D be the payload domain and $cnt_{\mathcal{C}}$ be the number of registered users.*
addCl.

$$\begin{aligned}
&\forall d \in D, \forall \mathbf{a} \in \{\bot, \mathsf{R}, \mathsf{RW}\}^{|\mathcal{DB}|}, \forall j \in [1, |\mathcal{DB}|]:\\
&\quad cap_i \leftarrow \mathsf{addCl}(cap_{\mathcal{O}}, \mathbf{a}) \implies\\
&\qquad (d \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle \iff d = \mathcal{DB}(j) \ \wedge \ \mathbf{a}(j) \neq \bot)\\
&\quad \wedge \ (\mathcal{DB}' \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle \iff \mathcal{DB}' = \mathcal{DB}[j \mapsto d] \ \wedge \ \mathbf{a}(j) = \mathsf{RW})
\end{aligned}$$

addE.

$$\begin{aligned}
&\forall d \in D, \forall \mathbf{a} \in \{\bot, \mathsf{R}, \mathsf{RW}\}^{|cnt_{\mathcal{C}}|}, \forall i \in [1, cnt_{\mathcal{C}}], \exists j:\\
&\quad \mathcal{DB}' \leftarrow \langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle \implies\\
&\qquad (|\mathcal{DB}'| = j) \wedge (|\mathcal{DB}| = j-1) \ \wedge \ (\mathcal{DB}' = \mathcal{DB}[j \mapsto d])\\
&\quad \wedge \ (d \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}')\rangle \iff d = \mathcal{DB}'(j) \ \wedge \ \mathbf{a}(i) \neq \bot)\\
&\quad \wedge \ \forall d'. \ (\mathcal{DB}'' \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d'), \mathcal{S}_{\mathsf{write}}(\mathcal{DB}')\rangle \iff \mathcal{DB}'' = \mathcal{DB}'[j \mapsto d'] \ \wedge \ \mathbf{a}(i) = \mathsf{RW})
\end{aligned}$$

chMode.

$$\begin{aligned}
&\forall d \in D, \forall \mathbf{a} \in \{\bot, \mathsf{R}, \mathsf{RW}\}^{cnt_{\mathcal{C}}}, \forall j \in [1, |\mathcal{DB}|], \forall i \in [1, cnt_{\mathcal{C}}]:\\
&\quad \langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle \implies\\
&\qquad (d \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle \iff d = \mathcal{DB}(j) \ \wedge \ \mathbf{a}(i) \neq \bot)\\
&\quad \wedge \ (\mathcal{DB}' \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle \iff \mathcal{DB}' = \mathcal{DB}[j \mapsto d] \ \wedge \ \mathbf{a}(i) = \mathsf{RW})
\end{aligned}$$

read.

$$\begin{aligned}
&\forall d \in D, \forall j \in [1, |\mathcal{DB}|], \forall i \in [1, cnt_{\mathcal{C}}]:\\
&\quad d \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle \implies d = \mathcal{DB}(j) \ \wedge \ \mathbf{AC}(i, j) \neq \bot
\end{aligned}$$

write.

$$\begin{aligned}
&\forall d \in D, \forall j \in [1, |\mathcal{DB}|], \forall i \in [1, cnt_{\mathcal{C}}]:\\
&\quad \mathcal{DB}' \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle \implies \mathcal{DB}' = \mathcal{DB}[j \mapsto d] \ \wedge \ \mathbf{AC}(i, j) = \mathsf{RW}
\end{aligned}$$

Theorem B.1 (Correctness)**.** *Let Π_{PE} and Π_{POE} be a predicate (resp. predicate-only) encryption scheme, Π_{PKE} and Π_{SE} be a public-key (resp. private-key) encryption scheme, and* ZKP *be a zero-knowledge proof system such that they all fulfill completeness. Then* GORAM *constructed in Section 2.6.3 satisfies the definition of correctness (Definition B.2).*

Proof sketch. The proof is conducted by protocol inspection on the implementation of each algorithm. Under the assumption that all of the encryption schemes, as well as the zero-knowledge proof system, are complete except with negligible probability, it directly follows from the analysis of the protocols that all of our instantiations fulfill the definition of correctness. □

B.4.2. Security Proofs

B.4.2.1. Proof of Theorem 2.4

Note that in our proofs we consider the adaptive version of each definition where the attacker is allowed to spawn and corrupt clients without restrictions. As a consequence our instantiation requires us to fix in advance the number of clients M supported by the construction. Alternatively, one could consider the selective versions of the security definitions where the attacker is required to commit in advance to the client subset that he wants to corrupt.

Proof of Theorem 2.4.1. Assume towards contradiction that there exists an adversary $\mathcal{A}$ that wins the secrecy game with probability non-negligibly greater than $1/2$ for some non-negligible function $\epsilon(\lambda)$, then we can construct the following reduction $\mathcal{B}$ against the CPA-security of Π_{PKE}. Note that the CPA-security notion that we consider allows the adversary to query one message pair (m_0, m_1) and to receive the encryption of m_b, depending on the random coin of the challenger, under polynomially-many *independent* public keys. Such an experiment can be proven to be equivalent to the textbook CPA-security notion by a standard hybrid argument. The reduction is elaborated below.
Simulation. $\mathcal{B}$ receives q-many public keys $(ek_1^*, \ldots, ek_q^*)$ and samples a string $\vec{m} \in \{0,1\}^M$ uniformly at random. For each client $i \in \{1, \ldots, M\}$, $\mathcal{B}$ fixes $ek_i^{\mathsf{CPA}} = ek_i^*$ if $m_i = 1$, otherwise it samples a key pair $(ek_i^{\mathsf{CPA}}, dk_i^{\mathsf{CPA}}) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}^{\mathsf{CPA}}(1^\lambda)$ and associates the client capability with ek_i^{CPA}. Afterwards the reduction simulates faithfully the operations queried by the adversary on the clients i such that $m_i = 0$. For the clients i where $m_i = 1$, the reduction performs the same steps as dictated by the protocol except that it skips the decryption procedure in $\mathcal{C}_{\mathsf{extData}}$. At some point of the execution the adversary outputs (d_0, d_1, j) and $\mathcal{B}$ forwards the pair (d_0, d_1) to the challenger, who replies with $(c_1^*, \ldots, c_q^*)$. The reduction writes $\mathbf{c}_{\mathsf{Data}}$ of entry j as follows:

$$c_{\mathsf{Data}}^i = \begin{cases} c_i^* & \text{if } m_i = 1 \\ \mathsf{E}(ek_i^{\mathsf{CPA}}, 0^{|d||vrs||\sigma|}) & \text{otherwise} \end{cases}$$

At some point of the execution the adversary returns a bit b' that the reduction forwards to the challenger.

Analysis. The reduction is obviously efficient. Assume that the string $\vec{m}$ denotes whether the client i is corrupted depending on the bit m_i. Then, whenever the reduction correctly guesses (event that we denote by GUESS) the subset of corrupted clients, the simulation perfectly reproduces the inputs that the adversary is expecting since the algorithm $\mathcal{C}_{\mathsf{extData}}$ does not produce any output visible to the server. Note that this event happens with probability at least 2^{-M}. Therefore, whenever the challenger coin is $b = 0$ then the reduction resembles the secrecy game for $b = 0$ and the same holds for $b = 1$. In particular,

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 0\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 0\right] \mathsf{Pr}\left[\mathsf{GUESS}\right]$$

and

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 1\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right] \mathsf{Pr}\left[\mathsf{GUESS}\right].$$

Whenever the string $\vec{m}$ is not correctly sampled (i.e., $\neg\mathsf{GUESS}$), then we can bind the success probability of the reduction to $1/2$. Therefore we have that

$$\begin{aligned} \left|\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 1\right]\right| &= \\ \left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid b = 1\right]\right| \cdot \mathsf{Pr}\left[\mathsf{GUESS}\right] &\geq \\ \epsilon(\lambda) \cdot 2^{-M}&. \end{aligned}$$

Since M is a fixed constant, this represents a contradiction to the CPA-security of Π_{PKE} and it concludes our proof. □

Proof of Theorem 2.4.2. The proof works by gradually modifying the experiment via game hops, in the following we provide the reader with an outline of our formal argument.

- $\mathsf{Exp}_0^{\mathcal{A}}(\lambda)$: Resembles exactly the integrity game.
- $\mathsf{Exp}_1^{\mathcal{A}}(\lambda)$: For each entry on which only non-corrupted clients have write permissions, we modify $\mathbf{c}_{\mathsf{Auth}}$ to encrypt $0^{|sk|}$.
- $\mathsf{Exp}_2^{\mathcal{A}}(\lambda)$: We substitute all the verification algorithms on signatures from the data owner with a default rejection if the data was not previously signed by the challenger itself. The challenger keeps track of the signed data via book-keeping.
- $\mathsf{Exp}_3^{\mathcal{A}}(\lambda)$: The honestly generated common reference string for the zero-knowledge proof system is substituted with the trapdoor common reference string.

The proof for each step is elaborated below. Note that we consider only the integrity of the main database and not of the personal stack of each client.

Claim: $\mathsf{Exp}_0^{\mathcal{A}}(\lambda) \approx \mathsf{Exp}_1^{\mathcal{A}}(\lambda)$. Assume towards contradiction that there exists an adversary $\mathcal{A}$ such that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_0^{\mathcal{A}}(\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_1^{\mathcal{A}}(\lambda)\right]\right| \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$. Then we can construct the a reduction $\mathcal{B}$ against the CCA-security of Π_{PKE}. Note that we allow the reduction to query polynomially-many message pairs under polynomially-many independent public keys, such a game is equivalent to the textbook notion of CCA-security by standard hybrid argument.

Simulation. $\mathcal{B}$ receives q-many public keys $(ek_1^*, \dots, ek_q^*)$ and samples a string $\vec{m} \in \{0,1\}^M$ uniformly at random. For each client $i \in \{1, \dots, M\}$ the reduction fixes $ek_i^{\mathsf{CCA}} = ek_i^*$ if $m_i = 1$, otherwise it samples a key pair $(ek_i^{\mathsf{CCA}}, dk_i^{\mathsf{CCA}}) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}^{\mathsf{CCA}}(1^\lambda)$ and associates the client capability with ek_i^{CCA}. The reduction simulates then all of the operations as specified in the protocol except that the calls of the decryption algorithm for ciphertext encrypted for keys of non-corrupted clients are substituted with queries to the decryption oracle provided by the challenger. Additionally, if an entry can be written only from clients such that for all i we have $m_i = 1$, then the reduction sends as many $(sk, 0^{|sk|})$ to the challenger, who replies with c_i^*. The entry $\mathbf{c}_{\mathsf{Auth}}$ is then constructed as follows:

$$c^i_{\mathsf{Auth}} = \begin{cases} c_i^* & \text{if } m_i = 1 \\ \mathsf{E}(ek_i^{\mathsf{CCA}}, 0^{|sk|}) & \text{otherwise} \end{cases}$$

At some point of the execution the adversary returns a bit b' that the reduction forwards to the challenger.

Analysis. The reduction is clearly efficient. Also whenever $\mathcal{B}$ guesses correctly the subset of corrupted clients (which we denote by GUESS), it is easy to see that it correctly resembles the inputs of $\mathsf{Exp}_0^{\mathcal{A}}$ when the coin of the challenger is $b = 0$ and $\mathsf{Exp}_1^{\mathcal{A}}$ otherwise. It follows that

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 0\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_0^{\mathcal{A}}(\lambda)\right] \mathsf{Pr}\left[\mathsf{GUESS}\right]$$

and

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 1\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_1^{\mathcal{A}}(\lambda)\right] \mathsf{Pr}\left[\mathsf{GUESS}\right].$$

Whenever the string $\vec{m}$ is not correctly sampled (i.e., $\neg\mathsf{GUESS}$), then we can bind the success probability of the reduction to 1/2. Therefore we can rewrite

$$|\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 1\right]| =$$
$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_0^{\mathcal{A}}(\lambda)\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_1^{\mathcal{A}}(\lambda)\right] \mathsf{Pr}\left[\mathsf{GUESS}\right]\right| \geq \epsilon(\lambda) \cdot 2^{-M} \quad .$$

Since M is a fixed constant, this represents a contradiction to the CCA-security of Π_{PKE}.

Claim: $\mathsf{Exp}_1^{\mathcal{A}}(\lambda) \approx \mathsf{Exp}_2^{\mathcal{A}}(\lambda)$**.** Assume towards contradiction that there exists an adversary such that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_1^{\mathcal{A}}(\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_2^{\mathcal{A}}(\lambda)\right]\right| \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$. Then we can construct a reduction $\mathcal{B}$ against the existential unforgeability of Π_{DS}.

Simulation. $\mathcal{B}$ takes as input the verification key vk and sets the verification key of the data owner to be $vk_{\mathcal{O}} = vk$. Then it guesses an index $i \in \{1, \dots, q\}$, where q is an upper-bound on the number of queries of the adversary to any interface, and starts the simulation of the game. The oracles are executed as specified in $\mathsf{Exp}_1^{\mathcal{A}}(\lambda)$ except that whenever the algorithm

sign is run on the secret key of the data owner on some message m_i, the reduction queries the oracle provided by the challenger instead. At the q-th query of the adversary the reduction parses the content of the query and checks whether it contains some valid pair $(m, \mathsf{sign}(sk_\mathcal{O}, m))$ such that m was not queried to the signing oracle, if this is the case than it returns $(m, \mathsf{sign}(sk_\mathcal{O}, m))$ to the challenger. The simulation of the adversary is interrupted after the q-th query.

Analysis. It is easy to see that the reduction is efficient and that it perfectly simulates the input that the adversary is expecting in $\mathsf{Exp}_1^\mathcal{A}(\lambda)$. We shall note that the only difference between $\mathsf{Exp}_1^\mathcal{A}(\lambda)$ and $\mathsf{Exp}_2^\mathcal{A}(\lambda)$ is when the adversary is able to output a message-signature pair that is valid under the verification key of the data owner such that the message was not signed by the challenger. We denote this event by FORGE. By assumption we have that

$$\mathsf{Pr}\left[\mathsf{FORGE}\right] = \left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_1^\mathcal{A}(\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_2^\mathcal{A}(\lambda)\right]\right| \geq \epsilon(\lambda).$$

Since the reduction guesses the query where FORGE happens with probability $\frac{1}{q}$ and the adversary cannot re-use any previously signed messages (due to the corresponding version number), $\mathcal{B}$ returns a valid forgery with probability $\frac{1}{q} \cdot \epsilon(\lambda)$, which is still non-negligible. Thus, we have derived a contradiction to the existential unforgeability of Π_{DS}.

Claim: $\mathsf{Exp}_2^\mathcal{A}(\lambda) \approx \mathsf{Exp}_3^\mathcal{A}(\lambda)$**.** Assume towards contradiction that there exists an adversary such that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_2^\mathcal{A}(\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_3^\mathcal{A}(\lambda)\right]\right| \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$. Then we can construct a reduction $\mathcal{B}$ against the extractability of ZKP.

Simulation. $\mathcal{B}$ receives either the honestly generated common reference string or the trapdoor one and plugs it in the public parameters of the scheme, the rest of the simulation proceeds as in $\mathsf{Exp}_2^\mathcal{A}(\lambda)$.

Analysis. It is easy to see that in the former case $\mathcal{B}$ perfectly simulates $\mathsf{Exp}_2^\mathcal{A}(\lambda)$ while in the latter it reproduces the inputs in $\mathsf{Exp}_3^\mathcal{A}(\lambda)$. Therefore the advantage of $\mathcal{A}$ carries over to $\mathcal{B}$, in particular

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B}(\mathsf{crs}) \mid \mathsf{crs} \leftarrow \{0,1\}^*\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_2^\mathcal{A}(\lambda)\right]$$

and

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B}(\mathsf{crs}) \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathcal{E}(1^\lambda)\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_3^\mathcal{A}(\lambda)\right].$$

Thus we have that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{R}(\mathsf{crs}) \mid \mathsf{crs} \leftarrow \{0,1\}^*\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{R}(\mathsf{crs}) \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathcal{E}(1^\lambda)\right]\right| \geq \epsilon(\lambda).$$

Which is a contradiction to the extractability of ZKP.

Claim: $\mathsf{Pr}\left[\mathsf{Exp}_3^\mathcal{A}(\lambda) = 1\right] \leq \mu(\lambda)$**.** In the previous steps we showed that for all adversaries $\mathsf{Exp}_0^\mathcal{A}(\lambda) \approx \mathsf{Exp}_3^\mathcal{A}(\lambda)$, therefore it is enough to show that the success probability in $\mathsf{Exp}_3^\mathcal{A}(\lambda)$

is negligible for any PPT machine $\mathcal{A}$. We note that the adversary can only modify the content of the main database through the $\mathcal{C}_{\mathsf{flush}}$ algorithm, therefore it is enough to argue about the integrity of this operation to bind the success probability of the adversary. Loosely speaking, for changing the content of an entry the adversary must prove the possession of a signing key, which means that any malicious attempt to circumvent the write access control would necessarily result in the disclosure of a hidden signing key. More formally, assume towards contradiction that there exists an adversary $\mathcal{A}$ such that

$$\Pr\left[\mathsf{Exp}_3^{\mathcal{A}}(\lambda) = 1\right] \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$, then we can build a reduction $\mathcal{B}$ against the existential unforgeability of Π_{DS}.

Simulation. $\mathcal{B}$ takes as input a verification key vk, then it guesses an index $i \in \{1, \ldots, q\}$, where q is an upper-bound on the number of queries of the adversary to the interface for adding entries and changing access permission, and an entry index $j \in \{1, \ldots, N\}$. When the q-th query happens, the reduction sets vk as the verification key associated to the target entry and continues the simulation. Anytime that a signature is required on some message m under the verification key vk, the reduction queries the signing oracle provided by the challenger instead. Afterwards, when the reduction executes $\mathcal{C}_{\mathsf{flush}}$ in interaction with the adversary, it runs the extractor $\mathcal{E}$ to obtain the witness w of the proof of knowledge produced by the adversary on the entry j. The reduction samples a message m not queried to the signing oracle yet and computes $\sigma \leftarrow \mathsf{sign}(w, m)$. The reduction returns (m, σ) to the challenger and interrupts the execution.

Analysis. $\mathcal{B}$ runs only polynomially bound algorithms, therefore it is efficient. Assume for the moment that the reduction correctly guesses the entry j that the adversary outputs in the challenge phase and the query q that last modifies the entry j in the query phase. Then we note that the simulation does not need to know the secret key associated to vk since it is never encrypted in $\mathbf{c}_{\mathsf{Auth}}$ when only non-corrupted clients have access to it (see hop 1). Also, it must be the case that the subsequent proofs of knowledge for entry j are computed against vk (if not they are automatically rejected, see hop 2). Therefore, due to the correctness of the extractor (that the reduction can execute from hop 3), we have that $\mathcal{B}$ extracts the valid key (and thus returns a valid forgery) with the same probability as the adversary returns a valid proof for a modified entry that no corrupted party has write access to. By assumption this happens with probability $\frac{1}{q} \cdot \frac{1}{N} \cdot \epsilon(\lambda)$, which is non-negligible in the security parameter. This is a contradiction to the existential unforgeability of Π_{DS}, so we can bind

$$\Pr\left[\mathsf{Exp}_3^{\mathcal{A}}(\lambda) = 1\right] \leq \mu(\lambda).$$

This concludes our proof. □

Proof of Theorem 2.4.3. The proof is developed by applying the same modifications to the experiment as outlined in the proof for Theorem 2.4.2 up to $\mathsf{Exp}_2^{\mathcal{A}}(\lambda)$. The indistinguishability arguments follow along the same lines. In the following we prove that the success probability of any adversary in the modified version of the tamper resistance game (which we denote by $\mathsf{Exp}_{\mathsf{t}}^{\mathcal{A}}(\lambda)$) is negligible in the security parameter. Since this game is indistinguishable from the original, the theorem holds true. Assume towards contradiction that there exists an adversary $\mathcal{A}$ such that

$$\mathsf{Pr}\left[\mathsf{Exp}_{\mathsf{t}}^{\mathcal{A}}(\lambda) = 1\right] \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$, then we can build a reduction $\mathcal{B}$ against the existential unforgeability of Π_{DS}.

Simulation. $\mathcal{B}$ takes as input a verification key vk, then it guesses an index $i \in \{1, \ldots, q\}$, where q is an upper-bound on the number of queries of the adversary to the interface for adding entries and changing access permission. When the q-th query happens, the reduction sets vk as the verification key associated to the target entry and continues the simulation. Anytime that a signature is required on some message m under the verification key vk, the reduction queries the signing oracle provided by the challenger instead. The rest of the simulation proceeds as specified in $\mathsf{Exp}_{\mathsf{t}}^{\mathcal{A}}(\lambda)$. When the adversary queries the challenge interface on the entry j, the reduction forwards the corresponding $(d||vrs, \sigma)$ to the challenger and interrupts the simulation.

Analysis. The reduction runs only polynomially bound algorithms, therefore it is efficient. We note that the simulation does not need to know the secret key associated with vk since it is never encrypted in $\mathbf{c}_{\mathsf{Auth}}$ when only non-corrupted clients have access to it (see hop 1). Also it must be the case that the validity of the entry j is checked against vk until another query to the interface for changing access permissions (if not the data is automatically rejected, see hop 2). Thus we have that whenever $\mathcal{B}$ successfully guesses the number of the last query that modifies the entry j that is later on sent from the adversary as a challenge, $\mathcal{B}$ returns a valid forgery with the same probability as the adversary returns a valid entry different from the one in the database maintained by the reduction. By assumption this happens with probability $\frac{1}{q} \cdot \epsilon(\lambda)$, which is non-negligible in the security parameter. This is a contradiction to the existential unforgeability of Π_{DS}, so we can bind

$$\mathsf{Pr}\left[\mathsf{Exp}_{\mathsf{t}}^{\mathcal{A}}(\lambda) = 1\right] \leq \mu(\lambda).$$

This concludes our proof. □

Proof of Theorem 2.4.4. The proof of obliviousness for the read protocol follows directly from the privacy of the Π_{PIR} scheme: it is easy to see that $\mathcal{C}_{\mathsf{extData}}$ is essentially a Π_{PIR} query, while $\mathcal{C}_{\mathsf{addDummy}}$ is completely independent from the index read. Arguing about the obliviousness of the write protocol requires a more careful analysis. In the following we gradually modify the experiment of obliviousness against malicious clients to obtain a simulation where the write algorithm is indistinguishable from the read (for the entries that the attacker is not allowed to access). The validity of the theorem follows. We hereby outline the modifications that we apply on the simulation:

- $\mathsf{Exp}_{0}^{\mathcal{A}}(\lambda)$: Resembles exactly the experiment of obliviousness against malicious client.

- $\mathsf{Exp}_1^{\mathcal{A}}(\lambda)$: We modify the public parameters to include a trapdoor common reference string, that is used later on by the challenger to simulate all of the zero-knowledge proofs during the execution of $\mathcal{C}_{\mathsf{flush}}$.
- $\mathsf{Exp}_2^{\mathcal{A}}(\lambda)$: We record all of the data signed with the key of the data owner in a list maintained by the challenger. We then substitute all the verification algorithms on signatures from the data owner with a default rejection if the data does not belong to the list.
- $\mathsf{Exp}_3^{\mathcal{A}}(\lambda)$: For all entries that no corrupted client can read, we change the $\mathcal{C}_{\mathsf{repl}}$ algorithm to compute $\mathbf{c}'_{\mathsf{Data}}$ and $\mathbf{c}_{\mathsf{BrCast}}$ as vectors of encryptions of 0.

Claim: $\mathsf{Exp}_0^{\mathcal{A}}(\lambda) \approx \mathsf{Exp}_1^{\mathcal{A}}(\lambda)$**.** Recall that the zero-knowledge of ZKP guarantees the existence of a simulator $\mathcal{S}$ that generates a common reference string crs and a trapdoor td such that crs is indistinguishable from an honestly generated reference string. The knowledge of td allows for simulating a proof for any statement without knowing the witness. Under these premises we can formally prove our claim. Assume towards contradiction that there exists an adversary $\mathcal{A}$ such that

$$\left|\Pr\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_0^{\mathcal{A}}(\lambda)\right] - \Pr\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_1^{\mathcal{A}}(\lambda)\right]\right| \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$. Then we can construct a reduction $\mathcal{B}$ against the zero-knowledge of ZKP.

Simulation. $\mathcal{B}$ plugs the common reference string crs in the public parameters of the scheme and starts the simulation as specified by the original protocols. Whenever the reduction has to compute a proof over a statement stmt_i and a witness w_i, it sends (stmt_i, w_i) to the challenger who replies with a non-interactive proof π_i. The reduction forwards π_i to the adversary and proceeds with the simulation. At some point of the execution the adversary returns a bit b' that the reduction forwards to the challenger.

Analysis. $\mathcal{B}$ is obviously efficient. Furthermore it is easy to see that whenever crs and the proofs π_i are honestly generated, the reduction perfectly simulates $\mathsf{Exp}_0^{\mathcal{A}}(\lambda)$, while when they are generated by the simulator $\mathcal{S}$, then the inputs are identically distributed as in $\mathsf{Exp}_1^{\mathcal{A}}(\lambda)$. It follows that

$$\Pr\left[1 \leftarrow \mathcal{B}(\mathsf{crs}) \mid \mathsf{crs} \leftarrow \{0,1\}^*\right] = \Pr\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_0^{\mathcal{A}}(\lambda)\right]$$

and

$$\Pr\left[1 \leftarrow \mathcal{B}(\mathsf{crs}) \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathcal{S}(1^\lambda)\right] = \Pr\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_1^{\mathcal{A}}(\lambda)\right].$$

By the initial assumption we have that

$$\left|\Pr\left[1 \leftarrow \mathcal{B}(\mathsf{crs}) \mid \mathsf{crs} \leftarrow \{0,1\}^*\right] - \Pr\left[1 \leftarrow \mathcal{B}(\mathsf{crs}) \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathcal{S}(1^\lambda)\right]\right| \geq \epsilon(\lambda).$$

This is a contradiction to the zero-knowledge of ZKP.

Claim: $\mathsf{Exp}_1^{\mathcal{A}}(\lambda) \approx \mathsf{Exp}_2^{\mathcal{A}}(\lambda)$. The proof for the indistinguishability of the two experiments follows along the same lines of the second step in the proof of Theorem 2.4.2.

Claim: $\mathsf{Exp}_2^{\mathcal{A}}(\lambda) \approx \mathsf{Exp}_3^{\mathcal{A}}(\lambda)$. Assume towards contradiction that there exists an adversary $\mathcal{A}$ such that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_2^{\mathcal{A}}(\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_3^{\mathcal{A}}(\lambda)\right]\right| \geq \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$. Then we can construct a reduction $\mathcal{B}$ against the CPA-security of Π_{PKE}. We stress that we allow the reduction to query polynomially-many message pairs under polynomially-many independent public keys; a standard hybrid argument is enough to show that this experiment is equivalent to the textbook notion of CPA-security.

Simulation. $\mathcal{B}$ receives q-many public keys $(ek_1^*, \ldots, ek_q^*)$ and samples a string $\vec{m} \in \{0,1\}^M$ uniformly at random. For each client $i \in \{1, \ldots, M\}$ the reduction fixes $ek_i^{\mathsf{CPA}} = ek_i^*$ if $m_i = 1$, otherwise it samples a key pair $(ek_i^{\mathsf{CPA}}, dk_i^{\mathsf{CPA}}) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}^{\mathsf{CPA}}(1^\lambda)$ and associates the client capability with ek_i^{CPA}. $\mathcal{B}$ simulates then the protocols as specified by the construction, ignoring the decryption procedure in the $\mathcal{C}_{\mathsf{extData}}$ algorithm (note that no output is displayed to the server anyway). Additionally, the $\mathcal{C}_{\mathsf{repl}}$ algorithm is modified as follows: for all entries that can only be read by clients i such that for all i we have $m_i = 1$, then the reduction sends to the challenger the tuples $(d'||vrs||\sigma, 0)$ and $(j||\ell||vrs, 0)$, where $(j, d', vrs, \sigma, \ell)$ are the index, the data, the version number, the signature, and the new index associated with the target entry. The challenger answers with $(c_{i,0}^*, c_{i,1}^*)$ and the reduction construct the vectors $\mathbf{c}'_{\mathsf{Data}}$ and $\mathbf{c}_{\mathsf{BrCast}}$ as follows:

$$c_{\mathsf{Data}}^{i,\prime} = \begin{cases} c_{i,0}^* & \text{if } m_i = 1 \\ \mathsf{E}(ek_i^{\mathsf{CPA}}, 0) & \text{otherwise} \end{cases}$$

and

$$c_{\mathsf{BrCast}}^{i} = \begin{cases} c_{i,1}^* & \text{if } m_i = 1 \\ \mathsf{E}(ek_i^{\mathsf{CPA}}, 0) & \text{otherwise} \end{cases}$$

At some point of the execution the adversary returns a bit b' that the reduction forwards to the challenger.

Analysis. As it runs only a polynomially bounded algorithm, $\mathcal{B}$ is clearly efficient. Assume for the moment that the reduction correctly guesses the subset of clients that the adversary is going to corrupt throughout the execution of the experiment. Then we argue that whenever the challenger samples $b = 0$ the reduction resembles the inputs that the adversary is expecting in $\mathsf{Exp}_2^{\mathcal{A}}(\lambda)$, while when $b = 1$ the reduction perfectly reproduces $\mathsf{Exp}_3^{\mathcal{A}}(\lambda)$. In order to see that we point out that the reduction does not need to know the randomness of $\mathbf{c}'_{\mathsf{Data}}$ to compute the proof as it is computed using the simulator and the trapdoor (see hop 1). Also, we observe that in $\mathsf{Exp}_2^{\mathcal{A}}(\lambda)$ any valid pair of vectors $\mathbf{c}'_{\mathsf{Data}}$ and $\mathbf{c}_{\mathsf{BrCast}}$ is always composed by encryptions under the initial set of public keys (otherwise they are automatically rejected, due to hop 2), therefore we can assess that the inputs that the reduction provides to the adversary are correctly distributed. It follows that whenever $\vec{m}$ is correctly sampled (GUESS) we have that

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 0\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_2^{\mathcal{A}}(\lambda)\right] \mathsf{Pr}\left[\mathsf{GUESS}\right]$$

and

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 1\right] \mathsf{Pr}\left[\mathsf{GUESS}\right] = \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_3^{\mathcal{A}}(\lambda)\right] \mathsf{Pr}\left[\mathsf{GUESS}\right].$$

Whenever the string $\vec{m}$ is not correctly sampled (i.e., $\neg\mathsf{GUESS}$), then we can bind the success probability of the reduction to $1/2$. Therefore we can rewrite

$$\begin{gathered} \left|\mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{B} \mid b = 1\right]\right| = \\ \left|\mathsf{Pr}\left[\mathsf{GUESS}\right]\left(\mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_2^{\mathcal{A}}(\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A} \mid \mathsf{Exp}_3^{\mathcal{A}}(\lambda)\right]\right)\right| \geq \epsilon(\lambda) \cdot 2^{-M} \end{gathered} .$$

Since M is a fixed constant, this represents a contradiction to the CPA-security of Π_{PKE}.

Claim: $\mathsf{Exp}_3^{\mathcal{A}}(\lambda) \approx negl(\lambda)$**.** The argument above shows that for all PPT adversaries $\mathcal{A}$ the original experiment for obliviousness against malicious clients is indistinguishable from $\mathsf{Exp}_3^{\mathcal{A}}(\lambda)$. It follows that the success probabilities in the two experiments must be the same (up to a negligible factor in the security parameter). Therefore in order to prove our theorem it is enough to observe that in $\mathsf{Exp}_3^{\mathcal{A}}(\lambda)$, for entries that no corrupted client can read, the algorithm $\mathcal{C}_{\mathsf{repl}}$ is identical to $\mathcal{C}_{\mathsf{addDummy}}$ and that the execution of $\mathcal{C}_{\mathsf{flush}}$ is completely independent of the content of the entries in the client's personal stack. This implies that in this context the read and write operations are indistinguishable, which we initially proved to be oblivious. Since obliviousness must hold only for entries that are not readable by corrupted clients (see Definition 2.6), the obliviousness of the construction follows as the advantage of any adversary is bound to a negligible function in the security parameter. This concludes our analysis. □

B.4.2.2. Proof of Theorem 2.5

It is easy to see that our modification only target the sequencer component shipped with TaoStore [170]. Then, obliviousness is inherited from the underlying ORAM scheme. Furthermore, secrecy and integrity rely on the sequencer modifications and are, thus, immediate.

B.4.2.3. Proof of Theorem 2.6

Proof of Theorem 2.6.1. We assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ that is able to break the secrecy game with non-negligible probability, namely:

$$\left|\mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A},\mathsf{secrecy}}^{\mathsf{GORAM}}(\lambda, 1) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A},\mathsf{secrecy}}^{\mathsf{GORAM}}(\lambda, 0) = 1\right]\right| \geq \epsilon(\lambda)$$

for some non-negligible $\epsilon(\lambda)$. Then we show that we can use such an adversary to build the following reduction $\mathcal{B}$ against the attribute-hiding property of the predicate-encryption scheme Π_{PE} defined in Definition A.8. The simulation is elaborated below.

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ from the challenger and it forwards it to $\mathcal{A}$. $\mathcal{B}$ initializes uniformly at random an attribute I and it sends to the challenger the tuple (I, I), who replies with the public key of the predicate-encryption scheme ppk^*. Finally $\mathcal{B}$ runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$ as described in Section 2.6 without $\mathsf{Gen}_{\mathsf{PE}}(1^\lambda)$ for the data part of the entries, setting $ppk_{\mathsf{Data}} = ppk^*$ instead. Subsequently $\mathcal{B}$ gives ek and $\mathcal{DB}$ to $\mathcal{A}$.

Queries. $\mathcal{B}$ simulates the oracles for $\mathcal{A}$ as specified in the definition and as described in Section 2.6, except for the following two oracles:

$\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$: $\mathcal{B}$ initializes a predicate f such that $f(I) = \perp$ and $\forall j \in \mathcal{DB}$ it holds that $f(I_j) = 0$ whenever $\mathbf{a}(j) = \perp$ and $f(I_j) = 1$ otherwise.

$\mathcal{O}_{\mathsf{corCl}}(i)$: $\mathcal{B}$ queries the oracle provided by the challenger on f_i so to retrieve the corresponding key $psk_{f_i}^{\mathsf{Data}}$. $\mathcal{B}$ constructs cap_i using such a key, which is handed over to $\mathcal{A}$.

Challenge. Finally, $\mathcal{A}$ outputs $(j, (d_0, d_1))$, where j is an index denoting the database entry on which $\mathcal{A}$ wishes to be challenged and (d_0, d_1) is a pair of entries such that $|d_0| = |d_1|$. $\mathcal{B}$ accepts the tuple only if $\mathbf{AC}(i, j) = \perp$, for every i corrupted by $\mathcal{A}$ in the query phase. $\mathcal{B}$ parses then the j-th entry as $E_j = (c_{1,j}, c_{2,j}, c_{3,j})$, it fetches $c_{\mathsf{Data}} \leftarrow \mathsf{D}(dk, c_{3,j})$, and it finally gets $d \leftarrow \mathsf{D}_{\mathsf{PE}}(psk_f^{\mathsf{Data}}, c_{\mathsf{Data}})$, for some suitable psk_f^{Data}. Afterwards, $\mathcal{B}$ sends the tuple $(m_0, m_1) = (d_0, d_1)$ to the attribute-hiding challenger. The challenger answers back with the challenge ciphertext $c^* \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk, I, m_b)$ where b is the internal bit of the attribute-hiding challenger. $\mathcal{B}$ uses c^* to execute $\langle \mathcal{C}_{\mathsf{write}}(cap_{\mathcal{O}}, j, d_b), \mathcal{S}_{\mathsf{write}}(\mathcal{DB}) \rangle$, computing the new entry in the following manner:

$$E'_j = \begin{pmatrix} c'_{1,j} \leftarrow \mathsf{Rnd}(ek, c_{1,j}, r_1) \\ c'_{2,j} \leftarrow \mathsf{E}(ek, \mathsf{R}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, c_{\mathsf{Auth}}, r_2)) \\ c'_{3,j} \leftarrow \mathsf{E}(ek, c^*) \end{pmatrix}.$$

Output. In the output phase $\mathcal{A}$ still has access to the oracles except for $\mathcal{O}_{\mathsf{addCl}}$ on input $\mathbf{a}$ such that $\mathbf{a}(j) \neq \perp$; $\mathcal{O}_{\mathsf{corCl}}$ on input i such that $\mathbf{AC}(i, j) \neq \perp$; and $\mathcal{O}^0_{\mathsf{chMode}}$ on input $\mathbf{a}, j$ with $\mathbf{a}(i) \neq \perp$ for some previously corrupted client i. Note that in case there exists some non-corrupted i such that $\mathbf{a}(i) \neq \perp$, $\mathcal{B}$ just simulates the $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{A}(\mathcal{DB}) \rangle$ protocol by rerandomizing the challenge ciphertext rather than re-encrypting it. Eventually, $\mathcal{A}$ stops, outputting a bit b'. $\mathcal{B}$ forwards b' to the attribute-hiding challenger.

Analysis. The simulation is clearly efficient, also it is easy to see that whenever the challenger samples $b = 0$, the simulation perfectly reproduces the inputs that $\mathcal{A}$ is expecting in $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, 0)$ and likewise for $b = 1$. The only difference is that in the output phase, the oracle $\mathcal{O}^0_{\mathsf{chMode}}$ on j is simulated with a rerandomization, rather than a re-encryption on c_{Data}. By definition of rerandomization, however, these two operations are indistinguishable to $\mathcal{A}$. Thus, we can state the following:

$$\Pr\left[\mathcal{B} \mapsto 1 | b = 1\right] \approx \Pr\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, 1) = 1\right].$$

and

$$\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 0\right] \approx \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, 0) = 1\right].$$

However, it follows from the initial assumption that

$$\left|\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 1\right] - \mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 0\right]\right| \geq \epsilon(\lambda),$$

which is clearly a contradiction with respect to the attribute-hiding property of the predicate encryption scheme Π_{PE}, and it proves the theorem.

□

Proof of Theorem 2.6.2. The proof is conducted by contradiction. Assume that there exists a PPT $\mathcal{A}$ that wins the experiment defined in Definition 2.3 with non-negligible probability. Then we construct an adversary $\mathcal{B}$ that breaks the soundness of the zero-knowledge proof system ZKP. The simulation works exactly as specified in Definition 2.3. Since the database is held on the side of $\mathcal{B}$, the adversary can only modify data by triggering oracles and all of the resulting operations are executed locally by $\mathcal{B}$ as described in Section 2.6.3 except for the read and write protocols. Whenever an operation affects the database $\mathcal{DB}$, $\mathcal{B}$ reflects these changes on $\mathcal{DB}'$, especially in the protocols executed due to the oracles $\mathcal{O}^1_{\mathsf{addE}}$, $\mathcal{O}^1_{\mathsf{chMode}}$, and $\mathcal{O}^1_{\mathsf{write}}$. This implies that the opportunity for the adversary to inject some data such that $\mathcal{B}$ does not update the local database $\mathcal{DB}'$ accordingly, is restricted to $\mathcal{O}^1_{\mathsf{read}}$ and $\mathcal{O}^1_{\mathsf{write}}$ for corrupted clients (since otherwise $\mathcal{B}$ executes those protocols locally without involving $\mathcal{A}$). We will briefly recall how the operations are performed from Section 2.6.3.

Read and write. When $\mathcal{A}$ triggers the read or write on a certain index j, $\mathcal{B}$ releases the path for the leaf l_j associated to such an entry $E = (E_1, \ldots, E_{b(D+1)})$, from ρ down to T_{D,l_j}. The database $\mathcal{DB}$ is kept blocked by $\mathcal{B}$ until $\mathcal{A}$ frees it again by submitting an updated path for l_j along with a valid shuffle proof. The proof looks as follows:

$$P = PK \left\{ \begin{array}{l} (\pi, r_1, \ldots, r_{b(D+1)}) : \\ \quad \forall \ell.\ E_\ell = \mathsf{Rnd}(ek, E_{\pi^{-1}(\ell)}, r_\ell) \end{array} \right\}. \tag{B.4}$$

It is easy to see that, by construction, the proof guarantees that the new path is just a shuffling and a re-randomization with respect to the old one. $\mathcal{B}$ verifies the proof against the new and the old path, i.e., it checks whether the proof of shuffle correctness verifies cryptographically, whether the inputs correspond to the components of the E_j, and whether the outputs correspond to the components of the E'_j. If all these checks succeed, $\mathcal{B}$ replaces the old with the new path. Subsequently $\mathcal{A}$ sends an updated top entry $E' = (c'_1, c'_2, c'_3)$ for an old top entry $E = (c_1, c_2, c_3)$ along with a proof of the decryption for the top entry of the updated path:

$$P_{\mathsf{Auth}} = PK \left\{ \left(psk^{\mathsf{Auth}}_f\right) :\ \mathsf{D}_{\mathsf{PE}}(psk^{\mathsf{Auth}}_f, c_{\mathsf{Auth}}) = 1 \right\} \tag{B.5}$$

together with information to access c_{Auth} (see Section 2.10.1 for how this information looks like in our concrete instantiation), and a proof that he did not change the index

$$P = PK \left\{ [\![c'_1]\!] = [\![c_1]\!] \right\}. \tag{B.6}$$

$\mathcal{B}$ checks then P_{Auth} against the content of c_2 of E and P against c_1 and c_1'. If all of the checks verify, $\mathcal{B}$ replaces the old by the new entry in the first position of the tree.

Analysis. We assume toward contradiction that there exists an adversary $\mathcal{A}$ such that the experiment $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{integrity}}(\lambda)$ outputs 1 with non-negligible probability, namely

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{integrity}}(\lambda) = 1\right] \geq \epsilon(\lambda).$$

As we argued before, by construction of the experiment we know that $\mathcal{A}$ can only inject entries by cheating in the $\mathcal{O}^1_{\mathsf{read}}$ and $\mathcal{O}^1_{\mathsf{write}}$ oracles. Thus we can restrict the success probability of $\mathcal{A}$ to the interaction within the $\mathcal{O}^1_{\mathsf{read}}$ and $\mathcal{O}^1_{\mathsf{write}}$ oracles. Furthermore we split the success probability over his ability of breaking the zero-knowledge proof system ZKP. We define BREAK to be the event in which $\mathcal{A}$ computes one zero-knowledge proof that verifies but without knowing the witness, i.e., he convinces the verifier of a false statement. In particular it follows from the initial assumption that

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{integrity}}(\lambda) = 1\right] &= \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \mathsf{BREAK}\right] \cdot \mathsf{Pr}\left[\mathsf{BREAK}\right] \\ &+ \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{BREAK}\right] \cdot \mathsf{Pr}\left[\neg\ \mathsf{BREAK}\right] \\ &\geq \epsilon(\lambda).\end{aligned}$$

It is easy to see that, given that $\mathcal{A}$ can compute a false statement that convinces the verifier in the zero-knowledge proof system ZKP, he can win the game with probability 1, e.g., writing on an entry which he does not have writing access to. Therefore

$$\mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \mathsf{BREAK}\right] = 1.$$

On the other hand, given that $\mathcal{A}$ does not break the soundness of the zero-knowledge proof system ZKP, the probability that $\mathcal{A}$ modifies an entry without $\mathcal{B}$ noticing is negligible in both read and write algorithms. This follows directly from the notion of correctness of the primitive Definition B.2. Thus we can rewrite

$$\mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{BREAK}\right] \leq negl(\lambda).$$

It follows that we can bind the success probability of $\mathcal{A}$ as

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{integrity}}(\lambda) = 1\right] &\approx 1 \cdot \mathsf{Pr}\left[\mathsf{BREAK}\right] + negl(\lambda) \cdot \mathsf{Pr}\left[\neg\ \mathsf{BREAK}\right] \\ &\approx \mathsf{Pr}\left[\mathsf{BREAK}\right] \geq \epsilon(\lambda).\end{aligned}$$

Since $\mathcal{A}$ can query the interfaces only a polynomial number of times, say q, $\mathcal{B}$ can simply store all of the transcripts of the queries to the $\mathcal{O}^1_{\mathsf{read}}$ and $\mathcal{O}^1_{\mathsf{write}}$ oracles and output one zero-knowledge proof chosen uniformly at random. There are at most $3 \cdot q$ zero-knowledge proof transcripts by construction of the protocol, therefore the probability that $\mathcal{B}$ outputs a zero-knowledge proof which verifies a false statement is lower bounded by $\epsilon(\lambda) \cdot \frac{1}{3 \cdot q}$ which is still a non-negligible value. This is clearly a contradiction with respect to the soundness property of the zero-knowledge proof system and it concludes our proof. □

Proof of Theorem 2.6.3. The proof is elaborated by contradiction, assuming we have a PPT adversary $\mathcal{A}$ such that he is able to break the security of the game with non-negligible probability, we build a distinguisher $\mathcal{B}$ such that it runs the aforementioned $\mathcal{A}$ as a black-box to break the attribute-hiding property of the underlying predicate encryption scheme Π_{PE}.

In particular, we assume toward contradiction that the following inequality holds:

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1\right] \geq \epsilon(\lambda).$$

Note that, by construction of the experiment, the challenge entry outputted by $\mathcal{A}$ is accepted by the challenger only if $\mathcal{A}$ never had read or write access to it. This also allows us to not track data modifications after a call to $\mathcal{O}^0_{\mathsf{chMode}}$ since whenever a corrupted client is able to modify the corresponding entry (the target of the permission change) then this entry and this client are excluded as the adversary's challenge. Hence, it is sufficient to only track changes in $\mathcal{DB}'$ that are originating from $\mathcal{O}^0_{\mathsf{addE}}$ and $\mathcal{O}^0_{\mathsf{write}}$. We now define a new event GUESS as the event in which $\mathcal{A}$ guesses the attribute encrypted in the c_{Data} cipher. It is easy to see that, with such a knowledge, $\mathcal{A}$ can always win the game by simply re-encrypting another payload d into c_{Data} of the j^*-th entry. It follows that when the challenger attempts to read the challenged index j^*, she will always succeed and she outputs 1 with overwhelming probability. Therefore we split the success probability of $\mathcal{A}$ over the probability that GUESS occurs:

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1\right] &= \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1 |\ \mathsf{GUESS}\right] \cdot \mathsf{Pr}\left[\mathsf{GUESS}\right] \\ &+ \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1 |\ \neg\ \mathsf{GUESS}\right] \cdot \mathsf{Pr}\left[\neg\ \mathsf{GUESS}\right].\end{aligned}$$

As reasoned above, the success probability of $\mathcal{A}$ given his knowledge of the attribute encrypted within c_{Data} is overwhelming, thus we can rewrite:

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1\right] &= \mathsf{Pr}\left[\mathsf{GUESS}\right] \\ &+ \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1 |\ \neg\ \mathsf{GUESS}\right] \cdot \mathsf{Pr}\left[\neg\ \mathsf{GUESS}\right].\end{aligned}$$

We now consider the success probability of $\mathcal{A}$ given that he does not know the attribute of the Π_{PE} cipher relative to the challenge entry. It follows from the proof of Theorem 2.6.2 that $\mathcal{A}$ cannot modify the cipher via the oracles provided by the challenger without being detected. However, since $\mathcal{A}$ stores $\mathcal{DB}$ locally, he can still perform local modifications. Nevertheless, in order to win the experiment, it must be the case that the challenger succeeds in reading j^*, that implies the challenge entry to be encrypted such that the capability held by the challenger allows her to access that entry. In practice, this means that the attribute I of the cipher c_{Data} in the challenge entry is orthogonal with respect to the predicate f held by the challenger, as defined in Definition A.7. By assumption $\mathcal{A}$ knows neither the attribute I in c_{Data} nor the predicate f, therefore the probability that he computes an attribute I' such that it is orthogonal to f is negligible by construction. Hence, it follows that the probability that $\mathcal{A}$ wins the experiment, given that he does not guess the attribute, is upper bounded by a negligible value. Then we have:

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1\right] = \mathsf{Pr}\left[\mathsf{GUESS}\right] + negl(\lambda) \cdot \mathsf{Pr}\left[\neg\ \mathsf{GUESS}\right],$$

which by assumption is lower bounded by:

$$\Pr\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1\right] \approx \Pr\left[\mathsf{GUESS}\right] \geq \epsilon(\lambda).$$

New Experiment. We now define the experiment $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}{}'(\lambda)$ analogously to the game $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda)$, except that in the challenge phase the adversary outputs the attribute I that is encrypted in the c_{Data} cipher of an entry, along with its challenge index j^*. By the argument above it follows that $\mathcal{A}$ has the same success probability in both games. The experiment is defined as follows, where we highlight changes with respect to $\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda)$ by solid boxes:

```
$\mathsf{Exp}^{\Pi_{\mathsf{GORAM}}}_{\mathcal{A},\mathsf{tam\text{-}res}}{}'(\lambda)$
$(cap_{\mathcal{O}}, \mathcal{DB}, \mathbf{AC}, \mathsf{CAP}, Cor, \mathcal{DB}') \leftarrow \mathsf{Setup}(1^\lambda, 1)$
$\mathbb{O} = \{\mathcal{O}_{\mathsf{addCl}}(\cdot), \mathcal{O}^0_{\mathsf{addE}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{chMode}}(\cdot,\cdot), \mathcal{O}_{\mathsf{corCl}}(\cdot),$
    $\mathcal{O}^0_{\mathsf{read}}(\cdot,\cdot), \mathcal{O}^0_{\mathsf{write}}(\cdot,\cdot,\cdot)\}$
// Assume that $\mathbf{AC}(i,j)$ stores a list of permissions
//   accounting for the history of permissions
//   for client $i$ on index $j$
$(j^*, \boxed{I^*}) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathcal{DB})$
if $\forall i, p.\ i \in Cor \wedge p \in \mathbf{AC}(i, j^*) \implies p \neq \mathsf{RW}$ then
  $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{A}(\mathcal{DB})\rangle$
  if $d^* \neq \mathcal{DB}'(j^*)$ $\boxed{\wedge I^* \text{ is the attribute associated to } j^*}$ then
    return 1
return 0
```

Reduction. Note that the experiment $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}{}'(\lambda)$ reproduces the experiment $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda)$ except that $\mathcal{A}$ must hold the knowledge of the attribute I associated to the challenge entry in order to win the former. However, since we argued that the success probability of $\mathcal{A}$ in the latter game implies such a knowledge, we can conclude that

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}(\lambda) = 1\right] &\approx \Pr\left[\mathsf{GUESS}\right] \\ &\approx \Pr\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{tam\text{-}res}}{}'(\lambda) = 1\right] \\ &\geq \epsilon(\lambda).\end{aligned}$$

Under this assumption, we can build the following reduction $\mathcal{B}$ against the attribute-hiding for multiple messages property of the predicate encryption scheme Π_{PE}. Note that, even though we did not explicitly state that property so far, it is implied by the attribute-hiding notion and we will subsequently show why this is the case. We define the experiment $\mathsf{Exp}^{\Pi_{\mathsf{PE}}}_{\mathcal{A},\mathsf{pe\text{-}multi}}(\lambda, b)$ as follows:

```
$\mathsf{Exp}^{\Pi_{\mathsf{PE}}}_{\mathcal{A},\mathsf{pe\text{-}multi}}(\lambda, b)$
$\Sigma^2 \ni (I_0, I_1) \leftarrow \mathcal{A}(1^\lambda)$
$(ppk, pmsk) \leftarrow \mathsf{Gen}_{\mathsf{PE}}(1^\lambda)$
// $\mathbb{O}$ on input $f_i \in \mathcal{F}$ by $\mathcal{A}$ outputs $pmsk_{f_i} \leftarrow \mathsf{K}_{\mathsf{PE}}(pmsk, f_i)$ if $f_i(I_0) = f_i(I_1)$
$((m_{0,0}, m_{1,0}), ..., (m_{0,t}, m_{1,t})) \leftarrow \mathcal{A}^{\mathbb{O}}(ppk)$
**if** $\forall 1 \leq i \leq t.\ |m_{0,j}| = |m_{1,j}| \wedge$
  $(\exists 1 \leq i \leq t.\ f_i(I_0) = f_i(I_1) = 1 \implies \forall 1 \leq j \leq t.\ m_{0,j} = m_{1,j})$ **then**
  $(c_1, \ldots, c_t) \leftarrow (\mathsf{E}_{\mathsf{PE}}(ppk, I_b, m_{b,0}), \ldots, \mathsf{E}_{\mathsf{PE}}(ppk, I_b, m_{b,1}))$
  // $\mathbb{O}'$ is defined as $\mathbb{O}$ with the additional restriction that
  // $f_i(I_0) = f_i(I_1) = 1 \implies \forall 1 \leq j \leq t.\ m_{0,j} = m_{1,j}$
  $b' \leftarrow \mathcal{A}^{\mathbb{O}'}(c_1, \ldots, c_t)$
  **if** $b' = b$ **then**
    **return** 1
**return** 0
```

The simulation works as follows:

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ from the challenger and it forwards it to $\mathcal{A}$. $\mathcal{B}$ initializes uniformly at random an attribute pair (I_0, I_1) and it sends it to the challenger, who replies with the public key of the predicate-encryption scheme ppk^*. Finally $\mathcal{B}$ runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$ as described in Section 2.6, but where $\mathsf{Gen}_{\mathsf{PE}}(1^\lambda)$ is used only to generate $(ppk_{\mathsf{Auth}}, pmsk_{\mathsf{Auth}})$, setting $ppk_{\mathsf{Data}} = ppk^*$ instead. Subsequently $\mathcal{B}$ gives ek to $\mathcal{A}$, and it initializes a second database $\mathcal{DB}'$ which is managed locally.

Queries. $\mathcal{B}$ then simulates the oracles provided to $\mathcal{A}$ as defined in the framework and as described in the protocol description, except the following oracles, which are implemented differently:

$\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$: $\mathcal{B}$ initializes a predicate f such that $\forall j \in \mathcal{DB}$ it holds that $f(I_j) = \mathbf{a}(j)$. Note that some entry j may be encrypted under either the I_0 or I_1 attribute, in this case f is chosen such that $f(I_0) = f(I_1) = \mathbf{a}(j)$. Then it queries the oracle provided by the challenger on f so to retrieve the corresponding key psk_f^{Data}. $\mathcal{B}$ constructs cap_i using that key, which is stored locally.

$\mathcal{O}^0_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ checks whether there exists some corrupted i such that $\mathbf{AC}(i, j) \neq \bot$, if this is the case it executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{A}(\mathcal{DB})\rangle$ in interaction with $\mathcal{A}$ who holds $\mathcal{DB}$. Otherwise, $\mathcal{B}$ sends the tuple $(m_0, m_1) = (d, d)$ to the challenger, who answers back with the challenge ciphertext $c^* \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk, I_b, m_b)$ that $\mathcal{B}$ uses to perform an execution of $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{A}(\mathcal{DB})\rangle$ in interaction with $\mathcal{A}$, setting $c_{\mathsf{Data}} \leftarrow c^*$. In both cases, $\mathcal{B}$ updates $\mathcal{DB}'$ with d at position j.

$\mathcal{O}^0_{\mathsf{chMode}}(\mathbf{a}, j)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{A}(\mathcal{DB})\rangle$ in interaction with $\mathcal{A}$, who holds $\mathcal{DB}$. Note that, if it is still the case that no corrupted i has access to the entry j, the c_{Data} of the respective entry is rerandomized during the execution of the protocol, rather than re-encrypted.

$\mathcal{O}_{\mathsf{corCl}}(i)$: $\mathcal{B}$ recomputes the predicate related to the i-th client in the access control matrix $\mathbf{AC}$ such that it fulfills the access control policy described by $\mathbf{AC}$. Note that some entry j may be encrypted under either the I_0 or I_1 attribute, in this case f is chosen such that $f(I_0) = f(I_1) = \mathbf{AC}(i,j)$. $\mathcal{B}$ then queries the oracle provided by the challenger on f and it receives back a secret key psk_f^{Data} which it sends to $\mathcal{A}$ together with the rest of the keys that form the capability cap_i.

Challenge. Finally, $\mathcal{A}$ outputs (j^*, I^*). $\mathcal{B}$ then checks whether $I^* = I_1$, if this is the case it outputs 1, otherwise it outputs 0.

It is easy to see that the reduction $\mathcal{B}$ is efficient and it perfectly simulates the inputs that $\mathcal{A}$ is expecting in the experiment $\mathsf{Exp}_{\mathcal{A},\mathsf{tam\text{-}res}}^{\mathsf{GORAM}\ \prime}(\lambda)$, therefore we can bind the success probability of $\mathcal{B}$ over the random choice of b in $\mathsf{Exp}_{\mathcal{B},\mathsf{pe\text{-}multi}}^{\Pi_{\mathsf{PE}}}$, to the success probability of $\mathcal{A}$. It follows from the initial assumption that

$$\mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{B},\mathsf{pe\text{-}multi}}^{\Pi_{\mathsf{PE}}}(\lambda, b) = 1\right] \approx \mathsf{Pr}\left[\mathsf{Exp}_{\mathcal{A},\mathsf{tam\text{-}res}}^{\mathsf{GORAM}\ \prime}(\lambda) = 1\right] \geq \epsilon(\lambda)$$

which is clearly a contradiction with respect the attribute-hiding for multiple messages property of the predicate encryption scheme Π_{PE}.

Attribute-hiding for multiple messages. What is left to show, is that the security notion of attribute-hiding of a predicate encryption scheme implies the attribute-hiding for multiple messages. The demonstration consists of a standard hybrid argument over the vector of ciphertexts. We define a hybrid distribution H_i where the first i ciphertexts contain the encryption of the message $m_{1,j}$ for $1 \leq j \leq i$ and the remaining $n - i$ are the encryption of $m_{0,j}$ for $n - i \leq j \leq n$. Observe that $H_0 = C_0$ and $H_n = C_1$. Assuming toward contradiction that there exists a distinguisher $\mathcal{A}$ such that:

$$|\mathsf{Pr}[\mathcal{A}(H_0) = 1] - \mathsf{Pr}[\mathcal{A}(H_1) = 1]| \geq \epsilon(\lambda),$$

then it must be the case that there exists some i such that

$$|\mathsf{Pr}[\mathcal{A}(H_i) = 1] - \mathsf{Pr}[\mathcal{A}(H_{i+1}) = 1]| \geq \epsilon(\lambda).$$

Notice that the only difference between H_i and H_{i+1} is that the $(i+1)$-th cipher is the encryption of $m_{0,i+1}$ in H_i, while it is the encryption of $m_{1,i+1}$ in H_{i+1}. Now, given a cipher c_b, it is easy to construct a distribution $H = (c_{1,1}, ..., c_{1,i}, c_b, c_{0,i+2}, ..., c_{0,n})$ which is equal to H_i if and only if c_b is the encryption of $m_{0,i+1}$ and it is equal to H_{i+1} otherwise. It follows that it is possible to distinguish the encryption of either $\vec{m}_0$ or $\vec{m}_1$ with the same probability with which $\mathcal{A}$ distinguishes H_i and H_{i+1}. This, however, contradicts our initial assumption and it proves the implication.

Versioning. Note that by the previous argument we only ruled out the cases where the server maliciously modifies an entry of the database without the client noticing it. However it is still possible for an adversary to win the game just by querying the write interface for an allowed modification on a given entry and then provide an old version of $\mathcal{DB}$ in the challenge phase. Note that this is a class of attacks that is inherent to the cloud storage design and can be prevented via standard techniques (e.g., by using gossip protocols [69] among the clients). For the sake of simplicity we do not consider such attacks in our proof and we implicitly assume that the adversary always runs the read algorithm on top of the most recent version of $\mathcal{DB}$ in the challenge phase of the cryptographic game. □

Proof of Theorem 2.6.4. The proof consists of the analysis of the distribution of the read and write operations over the access pattern of the paths in the binary tree. Combining the uniform distribution over the retrieved path with the indistinguishability among the read and write algorithms, it directly follows that any two sequences of queries are indistinguishable, i.e., it cannot exist any adversary who wins the experiment $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{obliv}}(\lambda, b)$ with non-negligible probability.

It follows from the design of the primitive that each bucket independently represents a trivial ORAM and therefore preserves oblivious access. Indeed, every bucket is always retrieved as a whole and it is re-randomized upon every access. By the CPA-security of the top layer public key encryption scheme Π_{PKE}, we can state that the rerandomization operation completely hides any modification of the data. Thus, we can restrict the information leaked by each operation to the path of the binary tree that gets retrieved and, in order to prove obliviousness, it is sufficient to demonstrate that each client's access leads to the same distribution over the choice of such a path.

Read. In the read algorithm the path associated with the target entry is initially retrieved by the client, note that the association leaf-entry was sampled uniformly at random. On the client-side, a new leaf-entry association is uniformly generated for the target entry and the path is arranged such that each entry is pushed as far as possible toward its designated leaf in the tree. In this process the client must make sure that a dummy entry is placed on top of the path, however the server does not gather any additional information from this procedure because of the CPA-security of Π_{PKE}. The path is then rerandomized and uploaded on the hosting server. It is easy to see that any further access of any entry (including the most recently accessed one) will always determine a path only depending on the association between leaf and entry, which is uniformly distributed. It follows that, for all accesses, the distribution over the choice of the path is uniform and independent with respect to the accessed entry. After the shuffling the client overwrites the dummy entry placed on top of the root node, however this does not affect the obliviousness of the algorithm since the memory location accessed is fixed.

Write. The write algorithm works analogously to the read, so the argument above still applies. The only difference in this scenario is that a target entry is selected to be placed on top of the root, instead of a dummy one. This, however, does not introduce any difference in the server's view since the choice of the entry to set on top of the path is again hidden by the CPA-security of Π_{PKE}. Thus, we achieve uniform distribution of the memory accesses in the write operation and consequently indistinguishability among the read and write protocols.

Zero-Knowledge. In both read and write cases, the client attaches along with the data sent to the server three non-interactive zero-knowledge proves that the server must verify in order to protect the integrity of the data. Specifically this set of proofs is composed of a proof of a shuffle correctness (line 10.11 and line 11.10), a proof of writing eligibility (line 10.14), and a proof of plaintext-equivalence for the index (line 10.16). However, due to the zero-knowledge property of ZKP the proved statements do not reveal any information so as to give $\mathcal{A}$ additional information that would allow him to break obliviousness. □

Proof of Theorem 2.6.5. The proof proceeds by contradiction. We show that, given an adversary who is able to win the experiment $\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{anonymity}}(\lambda, b)$ with non-negligible probability, we can construct an efficient algorithm that breaks the zero-knowledge property of ZKP.

By construction of the game, the only information available to the adversary in order to distinguish between the two capabilities is the execution of the read or write protocol, depending on the input he provides to the challenger. In both cases, it follows from the inspection of the protocol (see Section 2.6) that the only step which is not independent from the capability held by the client is the formulation of the authorization proof. We recall that the statement proves the knowledge of a key psk_f^{Auth} which can be used to successfully decrypt the ciphertext c_{Auth}. The proof looks as follows:

$$P_{\mathsf{Auth}} = PK\left\{\left(psk_f^{\mathsf{Auth}}\right) : \mathsf{D}_{\mathsf{PE}}(psk_f^{\mathsf{Auth}}, c_{\mathsf{Auth}}) = 1\right\},$$

note, indeed, that in the instantiation of our protocol, the capability of the client is implemented as the client's secret key psk_f^{Auth}. Thus, all of the read or write transcripts, except for the zero-knowledge proof, are trivially indistinguishable over the choice of the capability in case the two capabilities have the same access permissions on the challenge entry. Since the challenger does not generate the capabilities itself, we have to show that it can check the access permissions during the execution of the read or write protocol after the challenge so as to escape a trivial attack. The only thing necessary to check is that in case $d \neq \bot$, the challenger can decrypt P_{Auth} for the challenge entry using both capabilities. If that is not possible, it aborts the game and outputs 0. In case of read, it has to check whether there exists a dummy entry on the downloaded path such that it can decrypt P_{Auth} using both capabilities. If that is not the case, it also aborts the game and outputs 0. Hence, the challenger can always check the access control matrix equivalence for both capabilities, even though it does not know the full matrix. It follows that any information that the adversary gathers, can only be derived from the transcript of the authentication proof. Assuming toward contradiction that

$$\Pr\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{anonymity}}(\lambda, b) = 1\right] \geq \frac{1}{2} + \epsilon(\lambda),$$

then it must be the case that $\mathcal{A}$ is able to correctly guess the capability chosen by the challenger to formulate the proof P_{Auth} with advantage at least $\epsilon(\lambda)$ over the random choice of b. We define EXTRACT as the event in which $\mathcal{A}$ extracts some additional information from P_{Auth} about the capability used to construct the proof. It follows from the initial assumption that

$$\Pr\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{anonymity}}(\lambda, b) = 1\right] = \Pr\left[\mathcal{A} \text{ wins} \mid \mathsf{EXTRACT}\right] \cdot \Pr\left[\mathsf{EXTRACT}\right]$$

$$\begin{aligned}&+ \mathsf{Pr}\left[\mathcal{A} \text{ wins} \mid \neg\ \mathsf{EXTRACT}\right] \cdot \mathsf{Pr}\left[\neg\ \mathsf{EXTRACT}\right] \\ &\geq \frac{1}{2} + \epsilon(\lambda).\end{aligned}$$

It is easy to see that, given that $\mathcal{A}$ can deduce some information about the capability from P_{Auth}, i.e., that $\mathsf{EXTRACT}$ happens, he can win the game with probability 1. Therefore

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins} \mid \mathsf{EXTRACT}\right] = 1.$$

On the other hand, given that $\mathcal{A}$ does not break the zero-knowledge of P_{Auth}, the probability that $\mathcal{A}$ correctly guesses the capability sampled by the challenge is negligibly bigger than 1/2 in both read and write algorithms, as argued above. Thus we can rewrite

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins} \mid \neg\ \mathsf{EXTRACT}\right] \approx \frac{1}{2}.$$

It follows that we can bind the success probability of $\mathcal{A}$ as

$$\begin{aligned}&\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{GORAM}}_{\mathcal{A},\mathsf{anonymity}}(\lambda, b) = 1\right] \approx \\ &1 \cdot \mathsf{Pr}\left[\mathsf{EXTRACT}\right] + \tfrac{1}{2} \cdot \mathsf{Pr}\left[\neg\ \mathsf{EXTRACT}\right] \geq \tfrac{1}{2} + \epsilon(\lambda),\end{aligned}$$

then we have

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{EXTRACT}\right] + \frac{1}{2} \cdot (1 - \mathsf{Pr}\left[\mathsf{EXTRACT}\right]) &\geq \frac{1}{2} + \epsilon(\lambda) \\ \frac{1}{2} \cdot \mathsf{Pr}\left[\mathsf{EXTRACT}\right] &\geq \epsilon(\lambda) \\ \mathsf{Pr}\left[\mathsf{EXTRACT}\right] &\geq 2 \cdot \epsilon(\lambda),\end{aligned}$$

which is still a non-negligible value. This implies that $\mathcal{A}$ must be able to extract additional information from the proof without knowing the witnesses, which is clearly a contradiction with respect to the zero-knowledge property of ZKP and it concludes our proof. □

B.4.2.4. Proof of Theorem 2.7

Proof of Theorem 2.7.1. The proof works analogously to Theorem 2.6.1. □

Proof of Theorem 2.7.2. The proof is conducted by splitting the success probability of the adversary and showing that such probability is upper bounded by a sum of negligible values. We first define the event COLL as the event in which the adversary is able to find a collision on the tag t of the challenge entry of the experiment. We can express the probability that the adversary wins the experiment defined in Definition 2.8 as follows:

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{A-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] = \mathsf{Pr}\left[\mathcal{A} \text{ wins} \mid \mathsf{COLL}\right] \cdot \mathsf{Pr}\left[\mathsf{COLL}\right] + \mathsf{Pr}\left[\mathcal{A} \text{ wins} \mid \neg\ \mathsf{COLL}\right] \cdot \mathsf{Pr}\left[\neg\ \mathsf{COLL}\right].$$

It is easy to see that whenever the event COLL happens the adversary can easily win the game by arbitrarily modifying the entry and finding a collision for the hash tag. In the history of changes all of the versions of that entry will correctly verify and the challenger will not be able to blame any client in particular, thus making the adversary succeed with probability 1. By the above reasoning we can rewrite:

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{A-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] = \mathsf{Pr}\left[\mathsf{COLL}\right] + \mathsf{Pr}\left[\mathcal{A} \text{ wins} \mid \neg\ \mathsf{COLL}\right] \cdot \mathsf{Pr}\left[\neg\ \mathsf{COLL}\right].$$

We will now show that the probability that COLL happens is upper bounded by a negligible function. We again split the probability by defining the event $\mathsf{COLL}_{\mathsf{H}}$, which occurs when the adversary is able to find a collision in the standard hash function H, which is computed on $j \| c_{\mathsf{Auth}} \| cpk$. We obtain

$$\mathsf{Pr}\left[\mathsf{COLL}\right] = \mathsf{Pr}\left[\mathsf{COLL} \mid \mathsf{COLL}_{\mathsf{H}}\right] \cdot \mathsf{Pr}\left[\mathsf{COLL}_{\mathsf{H}}\right] + \mathsf{Pr}\left[\mathsf{COLL} \mid \neg\mathsf{COLL}_{\mathsf{H}}\right] \cdot \mathsf{Pr}\left[\neg\mathsf{COLL}_{\mathsf{H}}\right].$$

Clearly, if $\mathsf{COLL}_{\mathsf{H}}$ happens, then COLL occurs as well. Hence, we can rewrite

$$\mathsf{Pr}\left[\mathsf{COLL}\right] = \mathsf{Pr}\left[\mathsf{COLL}_{\mathsf{H}}\right] + \mathsf{Pr}\left[\mathsf{COLL} \mid \neg\mathsf{COLL}_{\mathsf{H}}\right] \cdot \mathsf{Pr}\left[\neg\mathsf{COLL}_{\mathsf{H}}\right].$$

It is apparent that $\mathsf{COLL}_{\mathsf{H}}$ can only occur with negligible probability due to the collision resistance of H. Hence, the event $\neg\mathsf{COLL}_{\mathsf{H}}$ must occur with overwhelming probability and we can again rewrite

$$\mathsf{Pr}\left[\mathsf{COLL}\right] \approx \mathsf{Pr}\left[\mathsf{COLL} \mid \neg\mathsf{COLL}_{\mathsf{H}}\right].$$

In order to prove that $\mathsf{Pr}\left[\mathsf{COLL} \mid \neg\mathsf{COLL}_{\mathsf{H}}\right]$ is negligible, we first define an intermediate game $\mathsf{Exp}^{\mathsf{A-GORAM}'}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$, we then show that such a game is indistinguishable from the original and finally we prove that in this latter experiment the probability of COLL, under the condition that $\mathsf{COLL}_{\mathsf{H}}$ does not happen, is negligible. It directly follows that the probability of COLL is also negligible in the original experiment $\mathsf{Exp}^{\mathsf{A-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$ since the two games are indistinguishable to the view of the adversary.

New Experiment. We define the intermediate game $\mathsf{Exp}^{\mathsf{A-GORAM}'}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$ as follows:

Setup. The challenger runs the Setup phase as in Definition 2.8. Additionally it sets a polynomial upper bound p on the number of queries to the interfaces $\mathcal{O}^1_{\mathsf{addE}}$ and $\mathcal{O}^1_{\mathsf{chMode}}$ and it picks a $q \in \{1...p\}$ uniformly at random.

Queries. The challenger runs the Query phase as in Definition 2.8, except for the following oracles, which are defined differently:

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: the challenger executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_w, s)$ where s is a random string such that $|s| = |csk|$.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: the challenger executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_w, s)$ where s is a random string such that $|s| = |csk|$. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

Challenge and output. The challenger runs the challenge and output phases as in Definition 2.8.

Claim: $\mathsf{Exp}_{\mathcal{A},\mathsf{acc\text{-}int}}^{\mathsf{A-GORAM}}(\lambda) \approx \mathsf{Exp}_{\mathcal{A},\mathsf{acc\text{-}int}}^{\mathsf{A-GORAM}'}(\lambda)$**.** We prove the claim with a reduction against the attribute-hiding property of the predicate-encryption scheme Π_{PE}. That is, given an adversary $\mathcal{A}$ that can efficiently distinguish the two games, we create a simulation $\mathcal{B}$ that breaks the attribute-hiding property with the same probability, thus such an adversary cannot exist. The reduction is depicted below.

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ from the challenger and it forwards it to $\mathcal{A}$. $\mathcal{B}$ initializes uniformly at random an attribute I and it sends to the challenger the tuple (I, I), who replies with the public key of the predicate-encryption scheme ppk^*. $\mathcal{B}$ then runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$, using $\mathsf{Gen}_{\mathsf{PE}}(1^\lambda)$ only to generate $(ppk_{\mathsf{Data}}, pmsk_{\mathsf{Data}})$, setting $ppk_{\mathsf{Auth}} = ppk^*$ instead. Subsequently $\mathcal{B}$ gives ek to $\mathcal{A}$. Finally, it sets a polynomial upper bound p on the number of queries to the interfaces $\mathcal{O}^1_{\mathsf{addE}}$ and $\mathcal{O}^1_{\mathsf{chMode}}$ and it picks a $q \in \{1...p\}$ uniformly at random.

Queries. $\mathcal{B}$ simulates the oracles as described, except for the following ones, which are simulated as described below:

$\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$: $\mathcal{B}$ initializes a predicate f such that $f(I) = \perp$ and $\forall j \in \mathcal{DB}$ it holds that $f(I_j) = 0$ whenever $\mathbf{a}(j) = \perp$ and $f(I_j) = 1$ otherwise.

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then $\mathcal{B}$ sends to the challenger the pair $(csk, s) = (m_0, m_1)$ where csk is the chameleon secret key relative to that entry and s is a random string such that $|s| = |csk|$. The challenger replies with $c^* \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, I, m_b)$ and $\mathcal{B}$ sets $c_{\mathsf{Auth}} = c^*$ for the target entry.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then $\mathcal{B}$ sends to the challenger the pair $(csk, s) = (m_0, m_1)$ where csk is the chameleon secret key relative to that entry and s is a random string such that $|s| = |csk|$. The challenger replies with $c^* \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, I, m_b)$ and $\mathcal{B}$ sets $c_{\mathsf{Auth}} = c^*$ for the target entry. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

$\mathcal{O}_{\mathsf{corCl}}(i)$: $\mathcal{B}$ queries the oracle provided by the challenger on the relative predicate f_i so to retrieve the corresponding key $psk^{\mathsf{Auth}}_{f_i}$. $\mathcal{B}$ constructs cap_i using that key, which is then handed over to $\mathcal{A}$.

Challenge. Finally, $\mathcal{A}$ outputs an index j^* which he wants to be challenged on. If there exists a capability cap_i provided to $\mathcal{A}$ such that $\mathbf{AC}(i, j^*) = \mathsf{RW}$, then $\mathcal{B}$ aborts. $\mathcal{B}$ runs $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ and $L \leftarrow \langle \mathsf{blame}(cap_{\mathcal{O}}, \mathsf{Log}, j^*)$ locally.

Output. $\mathcal{B}$ sends to $\mathcal{A}$ 1 if and only if $d^* \neq \mathcal{DB}'(j^*)$ and $\exists\ i \in L$ that has not been queried by $\mathcal{A}$ to the interface $\mathsf{corCl}(\cdot)$ or $L = []$. At any point of the execution $\mathcal{A}$ can output 0 or 1 depending on his guess about which game he is facing, $\mathcal{B}$ simply forwards the bit to the challenger and it stops the simulation.

Analysis. The simulation above it is clearly efficient, also it is easy to see that whenever the challenger samples its internal coin $b = 0$, the simulation of $\mathcal{B}$ perfectly reproduces $\mathsf{Exp}^{\mathsf{A-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$, thus:

$$\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 0\right] \approx \mathsf{Pr}\left[\mathcal{A} \mapsto 1 | b = 0\right].$$

Instead, whenever $b = 1$ the protocol executed by $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{A-GORAM}'}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$,

$$\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 1\right] \approx \mathsf{Pr}\left[\mathcal{A} \mapsto 1 | b = 1\right].$$

By our initial assumption $\mathcal{A}$ was able to distinguish between the two games with non-negligible probability, therefore the probability carries over

$$\left|\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 0\right] - \mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 1\right]\right| \geq \epsilon(\lambda),$$

which is clearly a contradiction to the attribute-hiding property of Π_{PE} and it proves our claim.

Claim: $\mathsf{Pr}\left[\mathsf{COLL} \mid \neg\mathsf{COLL}_{\mathsf{H}}\right]$ *in* $\mathsf{Exp}^{\mathsf{A-GORAM}'}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) \leq negl(\lambda)$**.** We demonstrate the claim via a reduction against the property of collision-resistance with key-exposure freeness of the chameleon hash function. Assume towards contradiction that there exists an adversary $\mathcal{A}$ such that the event COLL happens in $\mathsf{Exp}^{\mathsf{A-GORAM}'}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$ with non-negligible probability, we build the following algorithm $\mathcal{B}$ to efficiently break the collision-resistance with key-exposure freeness property:

Setup. $\mathcal{B}$ sets a polynomial upper bound p on the number of queries to the oracles $\mathcal{O}^1_{\mathsf{addE}}$ and $\mathcal{O}^1_{\mathsf{chMode}}$ and it picks a $q \in \{1...p\}$ uniformly at random. Then it runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$ and it hands over ek to $\mathcal{A}$.

Queries. $\mathcal{B}$ simulates the oracles provided to $\mathcal{A}$ as described in the definition and the construction, except the following ones:

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_w, s)$ where s is a random string such that $|s| = |csk|$.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_w, s)$ where s is a random string such that $|s| = |csk|$. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

Challenge. Finally, $\mathcal{A}$ outputs an index j^* which he wants to be challenged on. If there exists a capability cap_i provided to $\mathcal{A}$ such that $\mathbf{AC}(i, j^*) = \mathsf{RW}$, then the $\mathcal{B}$ aborts. It then runs $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ and $L \leftarrow \langle \mathsf{blame}(cap_{\mathcal{O}}, \mathsf{Log}, j^*)$ locally.

Output. $\mathcal{B}$ outputs 1 if and only if $d^* \neq \mathcal{DB}'(j^*)$ and $\exists\, i \in L$ that has not been queried by $\mathcal{A}$ to the interface $\mathsf{corCl}(\cdot)$ or $L = []$.

Analysis. The simulation is efficient and it perfectly reproduces the game that $\mathcal{A}$ is expecting. Note that by assumption we have that COLL happens with probability $\epsilon(\lambda)$, thus it must be the case that the adversary was able to compute a collision of the chameleon hash function in the challenge entry with non-negligible probability. Note that $\mathcal{B}$ selects the challenge entry for storing s in c_{Auth} with probability at least $\frac{1}{p}$. Thus, with probability at least $\frac{1}{p} \cdot \epsilon(\lambda)$ $\mathcal{A}$ was able to compute a collision without having any information on the secret key csk. The probability is still non-negligible, therefore this constitutes a contradiction to the collision resistance with key-exposure freeness of Π_{CH}. This proves our claim.

We have demonstrated that the event COLL does not occur with more than negligible probability, therefore we can rewrite the total success probability of the adversary as follows:

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathsf{A-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] &= negl(\lambda) + \Pr\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \cdot (1 - negl(\lambda)) \\ &\approx \Pr\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right].\end{aligned}$$

Thus, what is left to show is that the success probability of the adversary given that he is not able to compute a collision for Π_{CH}, is at most a negligible value in the security parameter. We do so by a reduction against the existential unforgeability of the digital signature scheme Π_{DS}. Assuming towards contradiction that there exists an adversary $\mathcal{A}$ such that $\Pr\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \geq \epsilon(\lambda)$ we can build a reduction $\mathcal{B}$ against the existential unforgeability $\mathsf{Exp}^{\Pi_{\mathsf{DS}}}_{\mathcal{A},\mathsf{euf}}$ of Π_{DS} as follows:

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ and the verification key vk^*. It runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$ setting $vk_{\mathcal{O}} = vk^*$ and it hands over ek to $\mathcal{A}$.

Queries. $\mathcal{B}$ simulates all oracles as specified in the definition and the construction except the following one, which is simulated differently:

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. In order to compute the correct signature on the respective chameleon hash tag t, $\mathcal{B}$ queries the signing oracle provided by the challenger and retrieves the signature tag σ.

Challenge. Finally, $\mathcal{A}$ outputs an index j^* which he wants to be challenged on. $\mathcal{B}$ parses the Log to search one version of that entry that contains a pair (t', σ') that has not been queried to the signing oracle and such that $\mathsf{vfy}(vk_{\mathcal{O}}, t', \sigma') = 1$.

Output. $\mathcal{B}$ outputs such a pair (t', σ') and it interrupts the simulation.

Analysis. The simulation is clearly efficient. It is easy to see that, in order to win the game, $\mathcal{A}$ must be able to change an entry without writing permission and by leaving it in a consistent state. This can be done by either computing a collision in the chameleon hash function or by forging a valid signature on the tag t. By assumption we ruled out the first hypothesis, thus the winning condition of the adversary implies the forgery of a verifying message-signature pair. Note that $\mathcal{A}$ could also just roll back to some previous version of the entry but this can be easily prevented by including some timestamp in the computation of the chameleon hash. The winning probability of the reduction then carries over:

$$\mathsf{Pr}\left[\mathcal{B}\ \mathsf{wins}\right] \approx \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \geq \epsilon(\lambda).$$

In this way we built an efficient adversary $\mathcal{B}$ that breaks the existential unforgeability of Π_{DS} with non-negligible probability, which is clearly a contradiction. Therefore it must hold that $\mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right]$ is a negligible function in the security parameter. Finally we have that:

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{A-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] \approx \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \leq negl(\lambda),$$

which concludes our proof. □

Proof of Theorem 2.7.3. The proof works analogously to Theorem 2.6.4. □

B.4.2.5. Proof of Theorem 2.8

Proof of Theorem 2.8.1. The proof is constructed by fixing the choice of the challenger over the sampling of the random coin and define intermediate hybrid games as follows

- $\mathsf{Exp}^{\mathcal{A}}_{1}(\lambda, b) = \mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, b)$
- $\mathsf{Exp}^{\mathcal{A}}_{2}(\lambda, b)$ is defined as $\mathsf{Exp}^{\mathcal{A}}_{1}(\lambda, b)$ with the difference that the key under which c_{Data} is encrypted is replaced by a random value rather than the key produced by broadcast encryption.

Then we show that the difference among any two neighboring games is bounded by a negligible value in the security parameter, therefore the advantage of the adversary in $\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, b)$ turns to be a sum of negligible values, which is still negligible. In particular we demonstrate the following:

$$\mathsf{Exp}^{\mathcal{A}}_{1}(\lambda, 0) \approx \mathsf{Exp}^{\mathcal{A}}_{2}(\lambda, 0) \approx \mathsf{Exp}^{\mathcal{A}}_{2}(\lambda, 1) \approx \mathsf{Exp}^{\mathcal{A}}_{1}(\lambda, 1)$$

Different entry structure. As opposed to the previous constructions, we have to slightly change the entry structure so as to compensate for the structural differences incurred by broadcast encryption. In particular, c_{Auth} has now the form $Hdr_{\mathsf{Auth}} \| \mathcal{E}(K_{\mathsf{Auth}}, csk)$ where $\langle Hdr_{\mathsf{Auth}}, K_{\mathsf{Auth}} \rangle \leftarrow \mathsf{E}_{\mathsf{BE}}(S_{\mathsf{Auth}}, bpk_{\mathsf{Auth}})$ and S_{Auth} is the set of clients with RW permissions. Likewise, c_{Data} has the form $Hdr_{\mathsf{Data}} \| \mathcal{E}(K_{\mathsf{Data}}, d)$ where $\langle Hdr_{\mathsf{Data}}, K_{\mathsf{Data}} \rangle \leftarrow \mathsf{E}_{\mathsf{BE}}(S_{\mathsf{Data}}, bpk_{\mathsf{Auth}})$ and S_{Data} is the set of clients with R permissions on the entry. Clients are equipped with private keys d_i that allow for decrypting as follows. If $i \in S_x$, then $K_x \leftarrow \mathsf{D}_{\mathsf{BE}}(S_x, i, d_i, Hdr_x, bpk_x)$, allowing to decrypt c_{Auth} if $x = \mathsf{Auth}$ and c_{Data} in case $x = \mathsf{Data}$.

New Experiment. We define $\mathsf{Exp}^{\mathcal{A}}_{2}(\lambda, b)$ for clarity:

Setup, query, and output phase. As defined in Definition 2.2.

Challenge. Finally, $\mathcal{A}$ outputs $(j,(d_0,d_1))$, where j is an index denoting the database entry on which $\mathcal{A}$ wishes to be challenged and (d_0,d_1) is a pair of entries such that $|d_0|=|d_1|$. The challenger accepts the request only if $\mathbf{AC}(i,j)=\bot$, for every i corrupted by $\mathcal{A}$ in the query phase. Afterwards, the challenger invokes $\langle \mathcal{C}_{\mathsf{write}}(cap_{\mathcal{O}},j,d_b),\mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle$ in interaction with $\mathcal{A}$, as explained in Section 2.8, with the difference that the new entry is computed as follows:

$$E'_j = \begin{pmatrix} c'_{1,j} \leftarrow \mathcal{E}(\mathcal{K},j) \\ c'_{2,j} \leftarrow \mathcal{E}(\mathcal{K},Hdr_{\mathsf{Auth}} \| \mathcal{E}(K_{\mathsf{Auth}},csk)) \\ c'_{3,j} \leftarrow \mathcal{E}(\mathcal{K},Hdr_{\mathsf{Data}} \| \mathcal{E}(K^*,d)) \end{pmatrix}$$

where $\langle Hdr_{\mathsf{Auth}},K_{\mathsf{Auth}}\rangle \leftarrow \mathsf{E}_{\mathsf{BE}}(S_{\mathsf{Auth}},bpk)$, $\langle Hdr_{\mathsf{Data}},K_{\mathsf{Data}}\rangle \leftarrow \mathsf{E}_{\mathsf{BE}}(S_{\mathsf{Data}},bpk)$, S_{Auth} (resp. S_{Data}) are the subset of users having RW (resp. R) access on the entry j, and K^* is a random string such that $|K^*|=|K_{\mathsf{Data}}|$ and in particular it is not necessarily the same key used to encrypt d.

Claim: $\mathsf{Exp}_1^{\mathcal{A}}(\lambda,b) \approx \mathsf{Exp}_2^{\mathcal{A}}(\lambda,b)$**.** We assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ that is able to distinguish among $\mathsf{Exp}_1^{\mathcal{A}}(\lambda,b)$ and $\mathsf{Exp}_2^{\mathcal{A}}(\lambda,b)$ with non-negligible probability, namely:

$$\left|\Pr\left[\mathsf{Exp}_1^{\mathcal{A}}(\lambda,b)=1\right]-\Pr\left[\mathsf{Exp}_2^{\mathcal{A}}(\lambda,b)=1\right]\right| \geq \epsilon(\lambda)$$

for some non-negligible $\epsilon(\lambda)$. Then we show that we can use such an adversary to build the following reduction $\mathcal{B}$ against the adaptive-security property of the broadcast encryption scheme Π_{BE} defined in Definition A.10. The simulation is elaborated below.

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ and the public key bpk^* from the challenger and it forwards 1^λ to $\mathcal{A}$. $\mathcal{B}$ runs $(cap_{\mathcal{O}},\mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$ as described in Section 2.8 using $\mathsf{Gen}_{\mathsf{BE}}(1^\lambda)$ only to setup $(bsk_{\mathsf{Auth}},bpk_{\mathsf{Auth}})$, setting $bpk_{\mathsf{Data}}=bpk^*$ instead. Finally $\mathcal{B}$ gives ek and $\mathcal{DB}$ to $\mathcal{A}$.

Queries. $\mathcal{B}$ simulates the oracles provided to $\mathcal{A}$ as defined and constructed, except for the following ones, which we describe below:

$\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$: $\mathcal{B}$ adds one client to the set S of clients.

$\mathcal{O}_{\mathsf{corCl}}(i)$: $\mathcal{B}$ queries the oracle provided by the challenger on i so to retrieve the corresponding key d_i. $\mathcal{B}$ constructs cap_i using such a key, which is handed over to $\mathcal{A}$.

Challenge. Finally, $\mathcal{A}$ outputs $(j,(d_0,d_1))$, where j is an index denoting the database entry on which $\mathcal{A}$ wishes to be challenged and (d_0,d_1) is a pair of entries such that $|d_0|=|d_1|$. $\mathcal{B}$ accepts the tuple only if $\mathbf{AC}(i,j)=\bot$, for every i corrupted by $\mathcal{A}$ in the query phase. $\mathcal{B}$ sets S^* to be the set of clients $\mathcal{C}_i$ for which $\mathbf{AC}(i,j)\neq\bot$ and it sends it to the challenger, who replies with the tuple (Hdr^*,K^*). $\mathcal{B}$ then executes $\langle \mathcal{C}_{\mathsf{write}}(cap_{\mathcal{O}},j,d_b),\mathcal{A}(\mathcal{DB})\rangle$, computing the new entry in the following manner:

$$E'_j = \begin{pmatrix} c'_{1,j} \leftarrow \mathcal{E}(\mathcal{K},j) \\ c'_{2,j} \leftarrow \mathcal{E}(\mathcal{K},Hdr_{\mathsf{Auth}} \| \mathcal{E}(K_{\mathsf{Auth}},csk)) \\ c'_{3,j} \leftarrow \mathcal{E}(\mathcal{K},Hdr^* \| \mathcal{E}(K^*,d_b)) \end{pmatrix}$$

where $c'_{1,j}$ and $c'_{2,j}$ are essentially re-encrypted from the retrieved entry, only $c'_{3,j}$ is freshly computed from the material sent by the challenger.

Output. In the output phase $\mathcal{A}$ still has access to the oracles except for $\mathcal{O}_{\mathsf{addCl}}$ on input $\mathbf{a}$ such that $\mathbf{a}(j) \neq \bot$; $\mathcal{O}_{\mathsf{corCl}}$ on input i such that $\mathbf{AC}(i, j) \neq \bot$; and $\mathcal{O}^0_{\mathsf{chMode}}$ on input $\mathbf{a}, j$ with $\mathbf{a}(i) \neq \bot$ for some previously corrupted client i. Note that in case there exists some non-corrupted i such that $\mathbf{a}(i) \neq \bot$, $\mathcal{B}$ just simulates the $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{A}(\mathcal{DB})\rangle$ protocol by rerandomizing the challenge ciphertext rather than re-encrypting it. Eventually, $\mathcal{A}$ stops, outputting a bit b', which $\mathcal{B}$ simply forwards to the challenger.

Analysis. The simulation is clearly efficient, also it is easy to see that whenever the challenger samples $b^* = 0$, i.e., the key K^* is the one generated together with Hdr^*, the simulation perfectly reproduces the inputs that $\mathcal{A}$ is expecting in $\mathsf{Exp}_1^{\mathcal{A}}(\lambda, b)$. The only difference is indeed that in the challenge phase when preparing the challenge entry, $c'_{\mathsf{Data}} = Hdr^*, \mathcal{E}(K^*, d)$ is generated by re-encryption in the simulation, while in the real experiment the header is constructed as a randomization of the old one; conversely, in the output phase, the oracle $\mathcal{O}^0_{\mathsf{chMode}}$ on j is simulated by a rerandomization on Hdr^*, rather than a re-encryption of d under a fresh header and key. By definition of rerandomization, however, these two operations are indistinguishable to $\mathcal{A}$. Note that we assume for simplicity the broadcast encryption scheme to be rerandomizable, however this feature is not strictly necessary since we could simulate the rerandomization by asking the challenger another challenge cipher. This does not affect the security of the scheme by standard hybrid argument. Thus, we can state the following:

$$\mathsf{Pr}\left[\mathcal{B} \mapsto 0 | b^* = 0\right] \approx \mathsf{Pr}\left[\mathsf{Exp}_1^{\mathcal{A}}(\lambda, b) = 1\right].$$

On the other hand, in case the challenger initializes $b^* = 1$, then $\mathcal{B}$ perfectly simulates the environment that $\mathcal{A}$ is expecting in $\mathsf{Exp}_2^{\mathcal{A}}(\lambda, b)$ since the key K^* under which d is encrypted is just a random key from the key space. Therefore we can assert that:

$$\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b^* = 1\right] \approx \mathsf{Pr}\left[\mathsf{Exp}_2^{\mathcal{A}}(\lambda, b) = 1\right].$$

However, it follows from the initial assumption that

$$\left|\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b^* = 1\right] - \mathsf{Pr}\left[\mathcal{B} \mapsto 0 | b^* = 0\right]\right| \geq \epsilon(\lambda),$$

which is clearly a contradiction with respect to the adaptive-security property of the broadcast encryption scheme Π_{BE}, and it proves the initial claim.

Claim: $\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 0) \approx \mathsf{Exp}_2^{\mathcal{A}}(\lambda, 1)$**.** We assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ that is able to distinguish among $\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 0)$ and $\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 1)$ with non-negligible probability, namely:

$$\left|\mathsf{Pr}\left[\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 0) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 1) = 1\right]\right| \geq \epsilon(\lambda)$$

For some non-negligible $\epsilon(\lambda)$. Then we show that we can we can use such an adversary to build the following reduction $\mathcal{B}$ against the CPA-security property of the private-key encryption scheme Π_{SE}. The simulation is elaborated below.

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ from the challenger and it forwards it to $\mathcal{A}$. $\mathcal{B}$ then runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$ as described in Section 2.8 and it gives ek and $\mathcal{DB}$ to $\mathcal{A}$.

Queries. $\mathcal{B}$ simulates the oracles provided to $\mathcal{A}$ as described in the definition and according to the construction.

Challenge. Finally, $\mathcal{A}$ outputs $(j, (d_0, d_1))$, where j is an index denoting the database entry on which $\mathcal{A}$ wishes to be challenged and (d_0, d_1) is a pair of entries such that $|d_0| = |d_1|$. $\mathcal{B}$ accepts the tuple only if $\mathbf{AC}(i, j) = \bot$, for every i corrupted by $\mathcal{A}$ in the query phase. $\mathcal{B}$ sends the tuple $(m_0, m_1) = (d_0, d_1)$ to the challenger, who answers back with the challenge ciphertext $c^* \leftarrow \mathcal{E}(k, d_b)$ that $\mathcal{B}$ uses to perform a local execution of $\langle \mathcal{C}_{\mathsf{write}}(cap_{\mathcal{O}}, j, d_b), \mathcal{A}(\mathcal{DB})\rangle$, computing the new entry in the following manner:

$$E'_j = \begin{pmatrix} c'_{1,j} \leftarrow \mathcal{E}(\mathcal{K}, j) \\ c'_{2,j} \leftarrow \mathcal{E}(\mathcal{K}, Hdr_{\mathsf{Auth}} \| \mathcal{E}(K_{\mathsf{Auth}}, csk)) \\ c'_{3,j} \leftarrow \mathcal{E}(\mathcal{K}, Hdr_{\mathsf{Data}} \| c^*) \end{pmatrix}.$$

Output. In the output phase $\mathcal{A}$ still has access to the interfaces except for $\mathcal{O}_{\mathsf{addCl}}$ on input $\mathbf{a}$ such that $\mathbf{a}(j) \neq \bot$; $\mathcal{O}_{\mathsf{corCl}}$ on input i such that $\mathbf{AC}(i, j) \neq \bot$; and $\mathcal{O}^0_{\mathsf{chMode}}$ on input $\mathbf{a}, j$ with $\mathbf{a}(i) \neq \bot$ for some previously corrupted client i. Eventually, $\mathcal{A}$ stops, outputting a bit b', which $\mathcal{B}$ forwards to the challenger.

Analysis. The simulation is obviously efficient, also it is easy to see that whenever the challenger samples $b = 0$, the simulation perfectly reproduces the inputs that $\mathcal{A}$ is expecting in $\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 0)$. Thus, we can state the following:

$$\mathsf{Pr}\left[\mathcal{B} \mapsto 0 | b = 0\right] \approx \mathsf{Pr}\left[\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 0) = 1\right].$$

On the other hand, in case the challenger initializes $b = 1$, then $\mathcal{B}$ perfectly simulates the environment that $\mathcal{A}$ is expecting in $\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 1)$. Therefore we can assert that:

$$\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 1\right] \approx \mathsf{Pr}\left[\mathsf{Exp}_2^{\mathcal{A}}(\lambda, 1) = 1\right].$$

However, it follows from the initial assumption that:

$$|\mathsf{Pr}\left[\mathcal{B} \mapsto 1 | b = 1\right] - \mathsf{Pr}\left[\mathcal{B} \mapsto 0 | b = 0\right]| \geq \epsilon(\lambda),$$

$$\left|\mathsf{Pr}\left[\mathsf{Exp}^{\Pi_{\mathsf{SE}}}_{\mathcal{A},\mathsf{cpa}}(\lambda, 1) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\Pi_{\mathsf{SE}}}_{\mathcal{A},\mathsf{cpa}}(\lambda, 0) = 1\right]\right| \geq \epsilon(\lambda),$$

which implies a non-negligible difference in the success probability of $\mathcal{B}$ with respect to the random choice of b and it clearly represents a contradiction with respect to the CPA-security property of the private-key encryption scheme Π_{SE}. This proves the initial claim.

$\mathsf{Exp}_1^{\mathcal{A}}(\lambda, 0) \approx \mathsf{Exp}_1^{\mathcal{A}}(\lambda, 1)$. By the previous claims it directly follows that the difference among each couple of neighboring games is bounded by a negligible value, thus the difference between $\mathsf{Exp}_1^{\mathcal{A}}(\lambda, 0)$ and $\mathsf{Exp}_1^{\mathcal{A}}(\lambda, 1)$ is a sum of negligible terms, which is, again, negligible. In particular

$$\mathsf{Exp}_1^{\mathcal{A}}(\lambda, 0) \approx \mathsf{Exp}_1^{\mathcal{A}}(\lambda, 1)$$

directly implies that:

$$\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, 0) \approx \mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{secrecy}}(\lambda, 1)$$

thus, for all PPT adversary the two experiments look indistinguishable. This concludes our proof. □

Proof of Theorem 2.8.2. The proof is conducted by splitting the success probability of the adversary and showing that such probability is upper bounded by a sum of negligible values. We first define the event COLL as the event in which the adversary is able to find a collision on the chameleon hash function Π_{CH} without knowing the secret key, for the challenge entry of the experiment. We can express the probability that the adversary wins the experiment defined in Definition 2.8 as follows:

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] &= \Pr\left[\mathcal{A}\ \mathsf{wins} \mid \mathsf{COLL}\right] \cdot \Pr\left[\mathsf{COLL}\right] \\ &\quad + \Pr\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \cdot \Pr\left[\neg\ \mathsf{COLL}\right].\end{aligned}$$

It is easy to see that whenever the event COLL happens the adversary can easily win the game by arbitrarily modifying the entry and computing a collision for the chameleon hash function. In the history of changes all of the versions of that entry will correctly verify and the challenger will not be able to blame any client in particular, thus making the adversary succeed with probability 1. By the above reasoning we can rewrite:

$$\Pr\left[\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] = \Pr\left[\mathsf{COLL}\right] + \Pr\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \cdot \Pr\left[\neg\ \mathsf{COLL}\right].$$

We will now show that the probability that COLL happens is upper bounded by a negligible function. We again split the probability by defining the event $\mathsf{COLL_H}$, which occurs when the adversary is able to find a collision in the standard hash function H, which is computed on $j \| c_{\mathsf{Auth}} \| cpk$. We obtain

$$\Pr\left[\mathsf{COLL}\right] = \Pr\left[\mathsf{COLL} \mid \mathsf{COLL_H}\right] \cdot \Pr\left[\mathsf{COLL_H}\right] + \Pr\left[\mathsf{COLL} \mid \neg\mathsf{COLL_H}\right] \cdot \Pr\left[\neg\mathsf{COLL_H}\right].$$

Clearly, if $\mathsf{COLL_H}$ happens, then COLL occurs as well. Hence, we can rewrite

$$\Pr\left[\mathsf{COLL}\right] = \Pr\left[\mathsf{COLL_H}\right] + \Pr\left[\mathsf{COLL} \mid \neg\mathsf{COLL_H}\right] \cdot \Pr\left[\neg\mathsf{COLL_H}\right].$$

It is apparent that $\mathsf{COLL_H}$ can only occur with negligible probability due to the collision resistance of H. Hence, the event $\neg\mathsf{COLL_H}$ must occur with overwhelming probability and we can again rewrite

$$\Pr\left[\mathsf{COLL}\right] \approx \Pr\left[\mathsf{COLL} \mid \neg\mathsf{COLL_H}\right].$$

In order to prove that $\Pr\left[\mathsf{COLL} \mid \neg\mathsf{COLL_H}\right]$ is negligible, we first define two intermediate games $\mathsf{Exp}^{\mathcal{A}}_1(\lambda)$ and $\mathsf{Exp}^{\mathcal{A}}_2(\lambda)$, we then show that these games are indistinguishable from the original and finally we prove that in this latter experiment the probability of COLL, under the condition that $\mathsf{COLL_H}$ does not happen, is negligible. It directly follows that the probability of COLL is also negligible in the original experiment $\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$ since the two games are indistinguishable to the view of the adversary.

New Experiment $\mathsf{Exp}^{\mathcal{A}}_1(\lambda)$**.** Intuitively, this new experiment is as the original one with the difference that the challenger at some point encrypts the chameleon hash secret key in some fixed predetermined query under a random key rather than the one output by the encryption procedure of the broadcast encryption scheme. We define the exact game $\mathsf{Exp}^{\mathcal{A}}_1(\lambda)$ as follows:

Setup:. The challenger runs the setup phase as in Definition 2.8. Additionally it sets a polynomial upper bound p on the number of queries to the interfaces $\mathcal{O}^1_{\mathsf{addE}}$ and $\mathcal{O}^1_{\mathsf{chMode}}$ and it picks a $q \in \{1...p\}$ uniformly at random.

Queries:. The challenger runs the query phase as in Definition 2.8, except for the following oracles:

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: the challenger executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} = Hdr_{\mathsf{Auth}} \| \mathcal{E}(K^*, csk_j)$ where K^* is a random key such that $|K^*| = |K_{\mathsf{Auth}}|$.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: the challenger executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} = Hdr_{\mathsf{Auth}} \| \mathcal{E}(K^*, csk_j)$ where K^* is a random key such that $|K^*| = |K_{\mathsf{Auth}}|$. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

Challenge and output. The challenger runs the challenge and output phases as in Definition 2.8.

New Experiment $\mathsf{Exp}^{\mathcal{A}}_2(\lambda)$**.** Intuitively, this new experiment is as the previous one with the difference that the challenger at some point does not encrypt the chameleon hash secret key, but rather some random key under a random key in some fixed predetermined query. We define the exact game $\mathsf{Exp}^{\mathcal{A}}_2(\lambda)$ as follows:

Setup:. The challenger runs the setup phase as in Definition 2.8. Additionally it sets a polynomial upper bound p on the number of queries to the interfaces $\mathcal{O}^1_{\mathsf{addE}}$ and $\mathcal{O}^1_{\mathsf{chMode}}$ and it picks a $q \in \{1...p\}$ uniformly at random.

Queries:. The challenger runs the query phase as in Definition 2.8, except for the following interfaces:

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: the challenger executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} = Hdr_{\mathsf{Auth}} \| \mathcal{E}(K^*, s)$ where K^* is a random key with $|K^*| = |K_{\mathsf{Auth}}$ and s is a random string such that $|s| = |csk_j|$.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: the challenger executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} = \mathit{Hdr}_{\mathsf{Auth}} \| \mathcal{E}(K^*, s)$ where K^* is a random key with $|K^*| = |K_{\mathsf{Auth}}|$ and s is a random string such that $|s| = |\mathit{csk}_j|$. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

Challenge and output. The challenger runs the challenge and output phases as in Definition 2.8.

Claim: $\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) \approx \mathsf{Exp}^{\mathcal{A}}_1(\lambda)$**.** We prove the claim with a reduction against the adaptive-security property of the broadcast encryption scheme Π_{BE}. That is, given an adversary $\mathcal{A}$ that can efficiently distinguish the two games, we create a simulation $\mathcal{B}$ that breaks the adaptive-security property with the same probability, thus such an adversary cannot exist. The reduction is depicted below.

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ and the public key bpk^* from the challenger and it forwards 1^λ to $\mathcal{A}$. $\mathcal{B}$ then runs $(\mathit{cap}_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$, using $\mathsf{Gen}_{\mathsf{BE}}(1^\lambda)$ only to generate $(\mathit{bsk}_{\mathsf{Data}}, \mathit{bpk}_{\mathsf{Data}})$, setting $\mathit{bpk}_{\mathsf{Auth}} = \mathit{bpk}^*$ instead. Subsequently $\mathcal{B}$ initializes an empty set S of clients and it gives ek to $\mathcal{A}$. Finally, it sets a polynomial upper bound p on the number of queries to the interfaces $\mathcal{O}^1_{\mathsf{addE}}$ and $\mathcal{O}^1_{\mathsf{chMode}}$ and it picks a $q \in \{1...p\}$ uniformly at random.

Queries. $\mathcal{B}$ simulates the oracles provided to $\mathcal{A}$ as described, except the following ones:

$\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$: $\mathcal{B}$ adds one client to the set S of clients.

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{addE}}(\mathit{cap}_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB}) \rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then $\mathcal{B}$ sends to the challenger the set S^* of clients having write access to the j-th entry. The challenger replies with the tuple (Hdr^*, K^*) and $\mathcal{B}$ sets $c_{\mathsf{Auth}} \leftarrow \mathit{Hdr}^* \| \mathcal{E}(K^*, \mathit{csk}_j)$ for the target j-th entry.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{chMode}}(\mathit{cap}_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB}) \rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then $\mathcal{B}$ sends to the challenger the set S^* of clients having write access to the j-th entry. The challenger replies with the tuple (Hdr^*, K^*) and $\mathcal{B}$ sets $c_{\mathsf{Auth}} \leftarrow \mathit{Hdr}^* \| \mathcal{E}(K^*, \mathit{csk}_j)$ for the target j-th entry. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

$\mathcal{O}_{\mathsf{corCl}}(i)$: $\mathcal{B}$ queries the oracle provided by the challenger on i so to retrieve the corresponding key bsk_i. $\mathcal{B}$ constructs cap_i using such a key, which is then handed over to $\mathcal{A}$.

Challenge. Finally, $\mathcal{A}$ outputs an index j^* which he wants to be challenged on. If there exists a capability cap_i provided to $\mathcal{A}$ such that $\mathbf{AC}(i, j^*) = \mathsf{RW}$, then $\mathcal{B}$ aborts. $\mathcal{B}$ runs $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(\mathit{cap}_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}) \rangle$ and $L \leftarrow \langle \mathsf{blame}(\mathit{cap}_{\mathcal{O}}, \mathsf{Log}, j^*) \rangle$ locally.

Output. $\mathcal{B}$ sends to $\mathcal{A}$ 1 if and only if $d^* \neq \mathcal{DB}'(j^*)$ and $\exists\ i \in L$ that has not been queried by $\mathcal{A}$ to the oracle $\mathcal{O}_{\mathsf{corCl}}(\cdot)$ or $L = []$. At any point of the execution $\mathcal{A}$ can output 0 or 1 depending on his guess about which game he is facing, $\mathcal{B}$ simply forwards such a bit to the challenger and it stops the simulation.

Analysis. The simulation above it is clearly efficient, also it is easy to see that whenever the challenger samples its internal coin $b = 0$, the simulation of $\mathcal{B}$ perfectly reproduces $\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda)$, thus:

$$\Pr[\mathcal{B} \mapsto 0 | b = 0] \approx \Pr[\mathcal{A} \mapsto 0 | b = 0].$$

Instead, whenever $b = 1$ the protocol executed by $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathcal{A}}_{1}(\lambda)$,

$$\Pr[\mathcal{B} \mapsto 1 | b = 1] \approx \Pr[\mathcal{A} \mapsto 1 | b = 1].$$

By our initial assumption $\mathcal{A}$ was able to distinguish between the two games with non-negligible probability, therefore the probability carries over

$$|\Pr[\mathcal{B} \mapsto 0 | b = 0] - \Pr[\mathcal{B} \mapsto 1 | b = 1]| \geq \epsilon(\lambda),$$

which is clearly a contradiction to the adaptive-security property of Π_{BE} and it proves our claim.

Claim: $\mathsf{Exp}^{\mathcal{A}}_{1}(\lambda) \approx \mathsf{Exp}^{\mathcal{A}}_{2}(\lambda)$**.** We prove the claim with a reduction against the CPA-security of the private-key encryption scheme Π_{SE}. That is, given an adversary $\mathcal{A}$ that can efficiently distinguish the two games, we create a simulation $\mathcal{B}$ that breaks the CPA game of Π_{SE} with the same probability, thus such an adversary cannot exist. The reduction is depicted below.

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ from the challenger and it forwards 1^λ to $\mathcal{A}$. $\mathcal{B}$ then runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$. Subsequently $\mathcal{B}$ initializes an empty set S of clients and it gives ek to $\mathcal{A}$. Finally, it sets a polynomial upper bound p on the number of queries to the interfaces $\mathcal{O}^1_{\mathsf{addE}}$ and $\mathcal{O}^1_{\mathsf{chMode}}$ and it picks a $q \in \{1...p\}$ uniformly at random.

Queries. $\mathcal{B}$ simulates the oracles provided to $\mathcal{A}$ as described, except the following ones:

$\mathcal{O}_{\mathsf{addCl}}(\mathbf{a})$: $\mathcal{B}$ adds one client to the set S of clients.

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then $\mathcal{B}$ sends to the challenger the message pair (csk_j, s) for some random value s fulfilling $|s| = |csk_j|$. The challenger answers with c^*, which $\mathcal{B}$ uses to construct $c_{\mathsf{Auth}} \leftarrow Hdr_{\mathsf{Auth}} || c^*$ for the target j-th entry.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then $\mathcal{B}$ sends to the challenger the message pair (csk_j, s) for some random value s fulfilling $|s| = |csk_j|$. The challenger answers with c^*, which $\mathcal{B}$ uses to construct $c_{\mathsf{Auth}} \leftarrow Hdr_{\mathsf{Auth}} || c^*$ for the target j-th entry. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

$\mathcal{O}_{\mathsf{corCl}}(i)$: $\mathcal{B}$ queries the oracle provided by the challenger on i so to retrieve the corresponding key bsk_i. $\mathcal{B}$ constructs cap_i using such a key, which is then handed over to $\mathcal{A}$.

Challenge. Finally, $\mathcal{A}$ outputs an index j^* which he wants to be challenged on. If there exists a capability cap_i provided to $\mathcal{A}$ such that $\mathbf{AC}(i, j^*) = \mathsf{RW}$, then $\mathcal{B}$ aborts. $\mathcal{B}$ runs $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ and $L \leftarrow \langle \mathsf{blame}(cap_{\mathcal{O}}, \mathsf{Log}, j^*)$ locally.

Output. $\mathcal{B}$ sends to $\mathcal{A}$ 1 if and only if $d^* \neq \mathcal{DB}'(j^*)$ and $\exists\ i \in L$ that has not been queried by $\mathcal{A}$ to the interface $\mathcal{O}_{\mathsf{corCl}}(\cdot)$ or $L = []$. At any point of the execution $\mathcal{A}$ can output 0 or 1 depending on his guess about which game he is facing, $\mathcal{B}$ simply forwards such a bit to the challenger and it stops the simulation.

Analysis. The simulation above it is clearly efficient, also it is easy to see that whenever the challenger samples its internal coin $b = 0$, the simulation of $\mathcal{B}$ perfectly reproduces $\mathsf{Exp}_1^{\mathcal{A}}(\lambda)$, thus:

$$\Pr[\mathcal{B} \mapsto 0 | b = 0] \approx \Pr[\mathcal{A} \mapsto 0 | b = 0].$$

Instead, whenever $b = 1$ the protocol executed by $\mathcal{B}$ perfectly simulates $\mathsf{Exp}_2^{\mathcal{A}}(\lambda)$,

$$\Pr[\mathcal{B} \mapsto 1 | b = 1] \approx \Pr[\mathcal{A} \mapsto 1 | b = 1].$$

By our initial assumption $\mathcal{A}$ was able to distinguish between the two games with non-negligible probability, therefore the probability carries over

$$|\Pr[\mathcal{B} \mapsto 0 | b = 0] - \Pr[\mathcal{B} \mapsto 1 | b = 1]| \geq \epsilon(\lambda),$$

which is clearly a contradiction to the CPA-security of Π_{SE} and it proves our claim.

Claim: $\Pr[\mathsf{COLL}]$ *in* $\mathsf{Exp}_2^{\mathcal{A}}(\lambda) \leq negl(\lambda)$**.** We demonstrate the claim via a reduction against the property of collision-resistance with key-exposure freeness of the chameleon hash function. Assume towards contradiction that there exists an adversary $\mathcal{A}$ such that the event COLL happens in $\mathsf{Exp}_2^{\mathcal{A}}(\lambda)$ with non-negligible probability, we build the following algorithm $\mathcal{B}$ to efficiently break the collision-resistance with key-exposure freeness property:

Setup. $\mathcal{B}$ sets a polynomial upper bound p on the number of queries to the interfaces addE and chMode and it picks a $q \in \{1...p\}$ uniformly at random. Then it runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^{\lambda})$ and it hands over ek to $\mathcal{A}$.

Queries. $\mathcal{B}$ simulates the oracles as described, except the following ones:

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} \leftarrow Hdr_{\mathsf{Auth}}, \mathcal{E}(K^*, s)$ where K^* and s are random strings such that $|K^*| = |K_{\mathsf{Auth}}|$ and $|s| = |csk_j|$.

$\mathcal{O}^1_{\mathsf{chMode}}(\mathbf{a}, j)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$ locally. If it holds that for all corrupted i and for the new entry index j, $\mathbf{AC}(i, j) \neq \mathsf{RW}$ and the query is the q-th query, then the new entry is computed with $c_{\mathsf{Auth}} \leftarrow Hdr_{\mathsf{Auth}}, \mathcal{E}(K^*, s)$ where K^* and s are random strings such that $|K^*| = |K_{\mathsf{Auth}}|$ and $|s| = |csk_j|$. On the other hand, if there exists a corrupted i such that $\mathbf{AC}(i, j) = \mathsf{RW}$, c_{Auth} is reverted to be consistent with the entry structure.

Challenge. Finally, $\mathcal{A}$ outputs an index j^* which he wants to be challenged on. If there exists a capability cap_i provided to $\mathcal{A}$ such that $\mathbf{AC}(i, j^*) = \mathsf{RW}$, then the $\mathcal{B}$ aborts. It then runs $d^* \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_{\mathcal{O}}, j^*), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$ and $L \leftarrow \langle \mathsf{blame}(cap_{\mathcal{O}}, \mathsf{Log}, j^*)$ locally.

Output. $\mathcal{B}$ outputs 1 if and only if $d^* \neq \mathcal{DB}'(j^*)$ and $\exists\ i \in L$ that has not been queried by $\mathcal{A}$ to the interface $\mathsf{corCl}(\cdot)$ or $L = []$.

The simulation is efficient and it perfectly reproduces the game that $\mathcal{A}$ is expecting. Note that by assumption we have that COLL happens with probability $\epsilon(\lambda)$, thus it must be the case that the adversary was able to compute a collision of the chameleon hash function in the challenge entry with non-negligible probability. Note that $\mathcal{B}$ selects the challenge entry for storing s encrypted under K^* in c_{Auth} with probability at least $\frac{1}{p}$. Thus, with probability at least $\frac{1}{p} \cdot \epsilon(\lambda)$ $\mathcal{A}$ was able to compute a collision without having any information on the secret key csk. The probability is still non-negligible, therefore this constitutes a contradiction to the collision resistance with key-exposure freeness of Π_{CH}. This proves our lemma.

We have demonstrated that the event COLL does not occur with more than negligible probability, therefore we can rewrite the total success probability of the adversary as follows:

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] &= negl(\lambda) + \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \cdot (1 - negl(\lambda)) \\ &\approx \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right].\end{aligned}$$

Thus, what is left to show is that the success probability of the adversary given that he is not able to compute a collision for Π_{CH}, is at most a negligible value in the security parameter. This is demonstrated through a reduction against the existential unforgeability of the digital signature scheme Π_{DS}. Assuming towards contradiction that there exists an adversary $\mathcal{A}$ such that $\mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \geq \epsilon(\lambda)$ we can build a reduction $\mathcal{B}$ against the existential unforgeability $\mathsf{Exp}^{\Pi_{\mathsf{DS}}}_{\mathcal{A},\mathsf{euf}}$ of Π_{DS} as follows:

Setup. $\mathcal{B}$ receives as input the security parameter 1^λ and the verification key vk^*. It runs $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda)$ setting $vk_{\mathcal{O}} = vk^*$ and it hands over ek to $\mathcal{A}$.

Queries. $\mathcal{B}$ simulates the oracles as described, except the following one:

$\mathcal{O}^1_{\mathsf{addE}}(\mathbf{a}, d)$: $\mathcal{B}$ executes $\langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$ locally. In order to compute the correct signature on the respective chameleon hash tag t, $\mathcal{B}$ queries the signing oracle provided by the challenger and retrieves the signature tag σ.

Challenge. Finally, $\mathcal{A}$ outputs an index j^* which he wants to be challenged on. $\mathcal{B}$ parses the Log to search one version of that entry that contains a pair (t', σ') that has not been queried to the signing oracle and such that $\mathsf{vfy}(vk_{\mathcal{O}}, t', \sigma') = 1$.

Output. $\mathcal{B}$ outputs such a pair (t', σ') and it interrupts the simulation.

Analysis. The simulation is clearly efficient. It is easy to see that, in order to win the game, $\mathcal{A}$ must be able to change an entry without writing permission and by leaving it in a consistent state. This can be done by either computing a collision in the chameleon hash function or by forging a valid signature on the tag t. By assumption we ruled out the first hypothesis, thus the winning condition of the adversary implies the forgery of a verifying message-signature pair. Note that the $\mathcal{A}$ could also just roll back to some previous version of the entry but this can be easily prevented by including some timestamp in the computation of the chameleon hash. The winning probability of the reduction then carries over:

$$\mathsf{Pr}\left[\mathcal{B}\ \mathsf{wins}\right] \approx \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \geq \epsilon(\lambda).$$

In this way we built an efficient adversary $\mathcal{B}$ that breaks the existential unforgeability of Π_{DS} with non negligible probability, which is clearly a contradiction. Therefore it must hold that $\mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right]$ is a negligible function in the security parameter. Finally we have that:

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{S-GORAM}}_{\mathcal{A},\mathsf{acc\text{-}int}}(\lambda) = 1\right] \approx \mathsf{Pr}\left[\mathcal{A}\ \mathsf{wins} \mid \neg\ \mathsf{COLL}\right] \leq negl(\lambda),$$

which concludes our proof. □

Proof of Theorem 2.8.3. The proof works analogously to Theorem 2.6.4. □

B.5. Algorithms for A-GORAM

Implementation of $(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^{\lambda}, n)$**.** Additionally to Algorithm 5 we include in the capability of the data owner $cap_{\mathcal{O}}$ the key pair $(vk_{\mathcal{O}}, sk_{\mathcal{O}}) \leftarrow \mathsf{Gen}_{\mathsf{DS}}(1^{\lambda})$, used for signing the tag t.

Implementation of $\{cap_i, \mathsf{deny}\} \leftarrow \mathsf{addCl}(cap_{\mathcal{O}}, \mathbf{a})$**.** No difference from Algorithm 6.

Implementation of $\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{addE}}(cap_{\mathcal{O}}, \mathbf{a}, d), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB})\rangle$**.** The difference from Algorithm 7 is line 7.10, where the new entry is instead composed of $E_k = \mathcal{E}(\mathcal{K}, j \| c^j_{\mathsf{Auth}} \| c^j_{\mathsf{Data}} \| cpk^j \| r^j \| t^j \| \sigma^j)$ where

$$\begin{aligned} &(cpk^j, csk^j) \leftarrow \mathsf{Gen}_{\mathsf{CHF}}(1^{\lambda}) && c^j_{\mathsf{Auth}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Auth}}, x_{w,j}, csk^j) \\ &r^j \leftarrow \{0,1\}^{\lambda} && c^j_{\mathsf{Data}} \leftarrow \mathsf{E}_{\mathsf{PE}}(ppk_{\mathsf{Data}}, x_{r,j}, d) \\ &t^j \leftarrow \mathsf{CH}(cpk^j, c^j_{\mathsf{Data}}, r^j) \| \mathsf{H}(j \| c^j_{\mathsf{Auth}} \| cpk^j) && \sigma^j \leftarrow \mathsf{sign}(sk_{\mathcal{O}}, t^j). \end{aligned} \tag{B.7}$$

Furthermore, the rerandomization in 7.13 is substituted by the re-encryption under the same symmetric key.

Eviction. The eviction algorithm is implemented as in Algorithm 8 with the difference that the proof P is never computed. Additionally, the rerandomization step (line 8.4) is substituted with a re-encryption of the top-level encryption layer.

Implementation of $\langle \mathcal{C}_{\mathsf{chMode}}(cap_{\mathcal{O}}, \mathbf{a}, j), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB})\rangle$**.** The algorithm is defined as in Algorithm 9, except for line 9.12 where the new entry is initialized as defined in Equation (B.7) (see addE above).

Implementation of $\{d, \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{read}}(cap_i, j), \mathcal{S}_{\mathsf{read}}(\mathcal{DB})\rangle$**.** The read algorithm follows Algorithm 10 until line 10.12 and stops there. Furthermore, the proof P is no longer computed.

Implementation of $\{\mathcal{DB}', \mathsf{deny}\} \leftarrow \langle \mathcal{C}_{\mathsf{write}}(cap_i, j, d), \mathcal{S}_{\mathsf{write}}(\mathcal{DB})\rangle$**.** The write algorithm is equivalent to the aforementioned read algorithm up to the point where we upload the entry E_k stored at the index j, which is modified as follows:

$$csk^j := \mathsf{D}_{\mathsf{PE}}(psk_{\mathsf{Auth}}, c^j_{\mathsf{Auth}}) \qquad d' := \mathsf{D}_{\mathsf{PE}}(psk_{\mathsf{Data}}, c^j_{\mathsf{Data}})$$

$$\hat{c}^j_{\mathsf{Data}} \leftarrow c^j_{\mathsf{Data}} \cdot d'^{-1} \cdot d \qquad \hat{r}^j \leftarrow \mathsf{Col}(csk^j, c^j_{\mathsf{Data}}, r^j, \hat{c}^j_{\mathsf{Data}})$$

$$E_k := \mathcal{E}(\mathcal{K}, j \| c^j_{\mathsf{Auth}} \| c^j_{\mathsf{Key}} \| \hat{c}^j_{\mathsf{Data}} \| \hat{r}^j \| cpk^j \| t^j \| \sigma^j)$$

C. Further Details of CRRP

C.1. Security Proofs for SRC

Proof of Theorem 4.5. Assume that there exists an adversary $\mathcal{A}$ that wins the EUF-CMA game for SRC with non-negligible probability for some integer n–the message vector length. Then we construct an adversary $\mathcal{B}$ against the EUF-CMA of the Pointcheval-Sanders signature scheme.

$\mathcal{B}$ sends n to the challenger, and receives as input from the challenger a verification key $vk = (X, Y_1, \ldots, Y_n) \in \mathbb{G}_2^{n+1}$. $\mathcal{B}$ in turn chooses $y_{n+1} \in \mathbb{Z}_q$, computes $Y_{n+1} = h^{y_{n+1}}$, and forwards $vk' = (X, Y_1, \ldots, Y_{n+1})$ to $\mathcal{A}$.

The query phase is simulated as follows:

$\sigma \leftarrow \mathsf{sign}(sk, (m_1, \ldots, m_n))$: $\mathcal{B}$ forwards $\mathcal{A}$'s message vector $(m_1, \ldots, m_n)$ to the challenger who answers with a signature σ, which $\mathcal{B}$ forwards to $\mathcal{A}$.

$(\sigma, oi, aux) \leftarrow \mathsf{sign_C}(sk, (m_1, \ldots, m_n), r)$: $\mathcal{B}$ forwards $(m_1, \ldots, m_n)$ as a query to the challenger who answers with $\sigma' = (\sigma_1, \sigma_2)$. $\mathcal{B}$ then computes $aux = \sigma_1^{y_{n+1}}$, $\sigma = (\sigma_1, \sigma_2 \cdot \sigma_1^{y_{n+1}r})$, and $oi = r$.

$(\sigma', oi, aux) \leftarrow \mathsf{comm}(sk, \sigma, r)$: $\mathcal{B}$ parses σ as (σ_1, σ_2) and outputs $\sigma' = (\sigma_1, \sigma_2 \cdot \sigma_1^{y_{n+1}r})$, $aux = \sigma_1^{y_{n+1}}$, and $oi = r$.

Since $\mathcal{A}$ wins with non-negligible probability, it must be the case that it outputs $(\sigma^*, (m_1^*, \ldots, m_n^*))$ with $\top = \mathsf{vfy}(\sigma^*, vk', (m_1^*, \ldots, m_n^*))$ or $(\sigma^*, oi^*, (m_1^*, \ldots, m_n^*))$ with $\top = \mathsf{vfy_C}(\sigma^*, vk', (m_1^*, \ldots, m_n^*), oi^*)$. In both cases, the tuple $(m_1^*, \ldots, m_n^*)$ has not been queried to any oracle, otherwise $\mathcal{A}$ would not win. In the first case, since the signature is not on a commitment, it must be the case that we also have $\top = \mathsf{vfy}(\sigma^*, vk, (m_1^*, \ldots, m_n^*))$ and hence, the tuple $(\sigma^*, (m_1^*, \ldots, m_n^*))$ is also a valid forgery to the Pointcheval-Sanders signature. $\mathcal{B}$ forwards $(\sigma^*, (m_1^*, \ldots, m_n^*))$ to the challenger. In the latter case, we have that

$$e(\sigma_1^*, X \cdot \prod_{i=1}^{n} Y_i^{m_i^*} \cdot Y_{n+1}^{oi^*}) = e(\sigma_2^*, h).$$

Rearranging terms, we get

$$e(\sigma_1^*, X \cdot \prod_{i=1}^{n} Y_i^{m_i^*}) \cdot e(\sigma_1^*, Y_{n+1}^{oi^*}) = e(\sigma_2^* \cdot ((\underbrace{\sigma_1^{*y_{n+1}}}_{:=aux^*})^{oi^*})^{-1}, h) \cdot e(\sigma_1^*, Y_{n+1}^{oi^*}).$$

Hence,

$$e(\sigma_1^*, X \cdot \prod_{i=1}^{n} Y_i^{m_i^*}) = e(\sigma_2^* \cdot (aux^{*oi^*})^{-1}, h).$$

This implies that $\sigma = (\sigma_1^*, \sigma_2^* \cdot (aux^{*oi^*})^{-1})$ is a valid forgery for the Pointcheval-Sanders scheme and $\mathcal{B}$ forwards $(\sigma, (m_1^*, \ldots, m_n^*))$ as a forgery.

To analyze, since $\mathcal{A}$ wins with non-negligible probability and since we have shown that $\mathcal{B}$ can turn any valid forgery of $\mathcal{A}$ into a valid forgery of the Pointcheval-Sanders scheme, we conclude that $\mathcal{B}$'s advantage is equivalent to $\mathcal{A}$'s advantage. Hence, $\mathcal{B}$ wins with non-negligible probability. This is a contradiction to our assumption. □

Proof of Theorem 4.6. We prove the claim directly via an information-theoretic argument. Since the oracle queries are answered independently of the challenge, i.e., using independent randomness for the opening information, we can ignore them and only consider the challenge.

Consider the challenger's response to the message challenge. Then we have

$$\sigma^* = (a^*, a^{*x + \sum_{i=1}^{n} y_i m_i^b + y_{n+1} r^*})$$

and $aux^* = a^{*r^*}$ where $a^* \in \mathbb{G}_1$ and $r^* \in \mathbb{Z}_q$ are chosen uniformly at random.

Leveraging an information-theoretic argument, the adversary learns, via σ^*, aux^*, and sk (even vk would be sufficient in the information-theoretic argument) the following quantities: $x, y_1, \ldots, y_{n+1}, x + \sum_{i=1}^{n} y_i m_i^b + y_{n+1} r^*$. Since the adversary does not know b, he can extract two different values for r^*, one for the $b = 0$ message vector and one for the $b = 1$ message vector. The adversary has, however, no way of validating which guess of r^* is the correct one. Each possibility is equally likely and hence, the adversary can only guess, which means that he has a zero advantage of determining b just based on the challenge response. □

C.2. Cryptographic Instantiations

For all cryptographic schemes, we stick to the notation introduced in Table 4.2. We recall in this section both concrete constructions that we deploy in our instantiation as well as the security definitions that we require.

Bilinear map setup. We instantiate the function BGSetup with the type-III pairings described by Miyaji et al. [147]. These curves allow for a secure setup of 112 bit security using secret key sizes of 224 bit.

Public-key encryption. We rely on two public-key encryption schemes with different properties, one being multiplicatively, the other being additively homomorphic. Any of the two schemes needs to fulfill the basic security notion IND-CPA.The randomization function Rnd is called secure if it does not affect the adversary's success probability in the above game more than negligible. Similarly, the components needed for distributed decryption, i.e., the functions GenEK, PartD, and DistD do not affect the game in the same

```
$(ek, dk) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$
    $x \leftarrow_{\$} \mathbb{Z}_q^*$
    return $(g^x, x)$

$ek \leftarrow \mathsf{GenEK}(ek_1, \ldots, ek_m)$
    return $\prod_{i=1}^m ek_i$

$c \leftarrow \mathsf{E}(ek, m)$
    $r \leftarrow_{\$} \mathbb{Z}_q^*$
    return $(g^r, m \cdot ek^r)$

$m \leftarrow \mathsf{D}(dk, c)$
    $(c_1, c_2) \leftarrow c$
    return $c_2 \cdot c_1^{-dk}$

$c' \leftarrow \mathsf{Rnd}(ek, c, r)$
    $(c_1, c_2) \leftarrow c$
    $(G, H) \leftarrow ek$
    $r \leftarrow_{\$} \mathbb{Z}_q^*$
    return $(c_1 \cdot G^r, c_2 \cdot H^r)$

$D_i \leftarrow \mathsf{PartD}(dk_i, c)$
    $(c_1, c_2) \leftarrow c$
    return $c_1^{dk_i}$

$m \leftarrow \mathsf{DistD}(D_1, \ldots, D_m, c)$
    $(c_1, c_2) \leftarrow c$
    return $c_2 \cdot \prod_{i=1}^m D_i^{-1}$

$(\{\top, \bot\}, \pi) \leftarrow \mathsf{PET}(dk_1, \ldots, dk_m, c, d)$
    for $1 \leq i \leq m$ do
        $s_i \leftarrow_{\$} \mathbb{Z}_q^*$
        $e_i = (c_1 d_1^{-1})^{s_i}$
        $f_i = (c_2 d_2^{-1})^{s_i}$
        $\pi_i = PK\left\{(\alpha) : \ e_i = (c_1 d_1^{-1})^\alpha \wedge f_i = (c_2 d_2^{-1})^\alpha\right\}$
    endfor
    $e = \prod_{i=1}^m e_i$
    $f = \prod_{i=1}^m f_i$
    for $1 \leq i \leq m$ do
        $D_i \leftarrow \mathsf{PartD}(dk_i, (e, d))$
        $\pi_{2i} \leftarrow PK\{(\alpha) : \ D_i \leftarrow \mathsf{PartD}(dk_i, (e, d))\}$
    endfor
    $y \leftarrow \mathsf{DistD}(D_1, \ldots, D_m, (e, d))$
    if $y = 1_{\mathbb{G}}$ then
        return $(\top, \pi_1, \ldots, \pi_{2m})$
    else
        return $(\bot, \pi_1, \ldots, \pi_{2m})$
    endif
```

Figure C.1.: The ElGamal encryption scheme with all necessary functionalities.

way. Furthermore, depending on the threshold t necessary to decrypt a ciphertext, any collection of less than t parties executing first PartD followed by DistD are not able to decrypt the message.

Concretely, we deploy the ElGamal encryption scheme [76] as it fulfills all properties that we require. Additionally to instantiating it for the elliptic curve setting in $\mathbb{G}_1$, we also instantiate it for ordinary groups $\mathbb{G}$ generated by $(p, \mathbb{G}, g) \leftarrow \mathsf{GSetup}(1^\lambda)$.

Independent of the setup, we briefly describe the algorithms that we use throughout the paper in Figure C.1 where $\mathbb{G}$ is a prime order group over $\mathbb{Z}_q$ with generator g.

Let us examine the correctness of the algorithms. We shall have that $m \leftarrow$

$\mathsf{D}(dk, \mathsf{E}(ek, m))$:

$$\mathsf{D}(dk, \underbrace{\mathsf{E}(ek, m)}_{=(c_1,c_2)=(g^r, m\cdot ek^r)}) = c_1^{-dk} c_2 = g^{-rx} m g^{rx} = m.$$

Moreover, it should hold that $m \leftarrow \mathsf{DistD}(D_1, \ldots, D_m, c)$ where $D_i \leftarrow \mathsf{PartD}(dk_i, c)$, $c \leftarrow \mathsf{E}(ek, m)$, and $ek \leftarrow \mathsf{GenEK}(ek_1, \ldots, ek_m)$:

$$\begin{aligned}
\mathsf{DistD}(D_1, \ldots, D_m, c) &= c_2 \cdot \prod_{i=1}^{m} D_i^{-1} \\
&= c_2 \cdot \prod_{i=1}^{m} c_1^{-dk_i} \\
&= m \cdot ek^r \cdot g^{-r \cdot \sum_{i=1}^{m} dk_i} \\
&= m \cdot (\prod_{i=1}^{m} ek_i)^r \cdot g^{-r \cdot \sum_{i=1}^{m} dk_i} \\
&= m \cdot g^{r \cdot \sum_{i=1}^{m} dk_i} \cdot g^{-r \cdot \sum_{i=1}^{m} dk_i} \\
&= m
\end{aligned}$$

Concerning the algebraic properties, ElGamal ciphertexts are multiplicatively homomorphic, which is easy to see: given $c_1 = (g^r, m_1 h^r)$ and $c_2 = (g^s, m_2 ek^s)$, the component-wise multiplication results in $(g^{r+s}, m_1 m_2 ek^{r+s})$. Given a ciphertext $c = (g^r, m \cdot ek^r)$ and a scalar $s \in \mathbb{Z}_q$, we have $c^s = (g^{rs}, m^s ek^{rs})$, i.e., an s-times multiplication of c with itself. We exploit both of these properties when computing, verifying, and decrypting identification tokens.

We also need an additively homomorphic encryption scheme that allows for distributed decryption. To this end, we deploy Paillier encryption [153], to which the techniques to distribute secret keys is similar to the one used in ElGamal. Paillier ciphertexts are additively homomorphic and allow for scalar multiplication.

We should also note that we sometimes have to use nested encryption, namely, when preparing the ballot. Here, the ciphertext $\hat{tkn}_i$ encrypts a ciphertext (the encrypted, blinded SSP of a user) for the service providers and the user proves knowledge of the underlying message. Since the used scheme is ElGamal, the user simply encrypts both parts of the ciphertext separately.

Service-specific pseudonyms. SSPs [143] enjoy two fundamental properties that we require for our instantiation of CR^{2}P, uniqueness and anonymity, which are defined in Definition A.15 and Definition A.16.

SSPs are compatible with ElGamal encryption, i.e., they lie in the message space of ElGamal and can, thus, be encrypted. In order to compute an SSP from an identity $x \in \mathbb{Z}_q^*$ and a service description $\mathcal{S} \in \mathbb{G}_1$, we compute $psd = S^x$ where $S := h(\mathcal{S})$. The hash function maps arbitrary strings to $\mathbb{G}_1$ (e.g., using Icart's technique [104]), i.e., the discrete logarithm of S to base g is unknown. In our protocols, we have to bind the pseudonym to the respective identity, which is originally blindly signed in the signature of the credential issuer. We do that in the proof P_i of a ballot B_i by showing that the discrete logarithms (the secret key part x of a user) of the first message in every credential part and the encrypted pseudonym are equivalent.

Zero-knowledge proofs. In the course of developing our instantiation, we have to generate several zero-knowledge proofs of knowledge, all of which we explicitly present in this section.

1. The designated verifier proof in CIReg:

$$\pi = DVP_{bid_i}\left\{\begin{array}{l} (\alpha, \beta_1, \ldots, \beta_\ell, \gamma_1, \ldots, \gamma_\ell) : \\ \quad \sigma_i^0 \leftarrow \mathsf{sign}(\alpha, (bid_i, r)) \wedge \\ \quad \forall 1 \leq j \leq \ell.\ (\sigma_i^j, aux_i^j, \beta_j) \leftarrow \mathsf{sign}_{\mathsf{C}}(\alpha, (bid_i, r, at_i^j), \gamma_j) \wedge \\ \qquad (j \in I \implies (\overline{oi}_i^{\mathsf{SP}})_j = \mathsf{E}^+(ek_{\mathsf{SP}}, \beta_j)) \end{array}\right\}.$$

Concretely, using identities in $\mathbb{Z}_q^*$ and blinded identities in $\mathbb{G}$ defined as $bid \leftarrow g^{id}$ for a group generator g, our instantiation of SRCs over the same group $\mathbb{G}$, and Paillier encryption [153] for the additively homomorphic encryption of the opening information where we use the public key (G, N) (G a generator and N an RSA modulus), the proof, in standard PK notation using the designated verifier technique based on Jakobsson et al. [111] becomes

$$\pi = PK\left\{\begin{array}{l} (\beta_1, \ldots, \beta_\ell, \{\eta_j\}_{j\in I}, \_) : \\ \quad \forall 1 \leq j \leq \ell.\ \hat{\sigma}_i^j = e(\sigma_{i,1}^j, Y_4)^{\beta_j} \wedge \\ \quad \forall j \in I.\ (\overline{oi}_i^{\mathsf{SP}})_j = G^{\beta_j} \cdot \eta_j^N \mod N^2 \\ \vee \\ \quad bid_i = g^{\xi} \end{array}\right\}$$

where the wild-card stands for the unknown identity ξ and $\sigma_{i,1}^j = g^{\alpha_i^j}$ and $\hat{\sigma}_i^j = e(\sigma_{i,2}^j, h) \cdot e(bid_i^{\alpha_i^j}, Y_1)^{-1} \cdot e(\sigma_{i,1}^j, Y_2)^{-r} \cdot e(\sigma_{i,1}^j, Y_3)^{-at_i^j} = e(\sigma_{i,1}^j, Y_4)^{r_j}$. If the user is to fake this proof, she will leave out all witnesses except ξ to generate the proof, so the right disjunct (the last row) is true while the rest is faked.

We implement the different statements as follows: the proof for correct signature generation is nothing but a proof of knowledge of representations of discrete logarithms paired with equality of discrete logarithm proofs [40, 59]. The proof of plaintext knowledge is also a combination of discrete logarithm equality and a zero-knowledge argument of knowledge of plaintext knowledge due to Groth [93]. Here we have to be careful when implementing the scheme since the proofs are setup in different groups, so we have to make sure that the message space of the Paillier cryptosystem can fit any possible opening information. Furthermore, later on we homomorphically add a randomization to the opening information, which must be compensated by the setup since otherwise it can no longer be decrypted after the homomorphic operation. Hence, it must be the case that $N > 2q$, so as to fit the full sum of two opening information, each in $\mathbb{Z}_q^*$. Finally, the proof of knowing the private identity is a standard Schnorr proof [173]. Everything is combined using an OR-proof [58].

2. The proof P_j^1 and P_j^2 in SPReg of the forms

$$P_j^1 = PK\left\{(\alpha):\ \mathit{bid}_i \leftarrow \mathsf{blind}(\alpha)\ \wedge\ \top = \mathsf{vfy}(\sigma_{\mathit{id}_i}, \mathit{vk}_{\mathsf{PKI}}, \alpha)\right\}$$

and

$$P_j^2 = DVP_{\mathit{bid}_i}\left\{(\gamma) : B_j = g^\gamma\ \wedge\ \mathit{tkn}' = \mathit{tkn}^\gamma\right\}.$$

To express P_j^1 in concrete form, we usean SSP with $\mathcal{S}$ describing the service. The SRC need not be hidden, thus the proof in concrete form becomes

$$P_j^1 = PK\left\{(\alpha):\ \mathit{bid}_i = g^\alpha\ \wedge\ e(\sigma_{\mathsf{PKI},1}, X_{\mathsf{PKI}}) \cdot e(\sigma_{\mathsf{PKI},1}, Y_{\mathsf{PKI}})^\alpha = e(\sigma_{\mathsf{PKI},2}, h)\right\}.$$

That proof can be instantiated using the techniques mentioned before.

For P_j^2, we have

$$P_j^2 = PK\left\{\begin{array}{l} (\gamma, \_) : \\ \quad B_j = g^\gamma\ \wedge \\ \quad \mathit{tkn}' = (\mathit{tkn}_1', \mathit{tkn}_2') = (\mathit{tkn}_1^\gamma, \mathit{tkn}_2^\gamma) = \mathit{tkn}^\gamma \\ \vee \\ \quad \mathit{bid}_i = g^\xi \end{array}\right\}$$

which also does not require new proof techniques.

3. The proof P_i in Rate

$$P_i = PK\left\{\begin{array}{l} (\sigma_i^0, \{\tilde{\sigma}_i^j\}_{j\in I}, \mathit{id}, r, \mathit{tkn}_i, v, \sigma_{\mathit{id}}) : \\ \quad \top = \mathsf{vfy}(\sigma_{\mathit{id}}, \mathit{vk}_{\mathsf{PKI}}, \mathit{id})\ \wedge\ \top = \mathsf{vfy}(\sigma_i^0, \mathit{vk}_{\mathsf{CI}}, (\mathit{id}, r))\ \wedge \\ \quad \forall j \in I.\ \hat{\sigma}_i^j = \mathsf{vfy}_{\mathsf{start}}(\tilde{\sigma}_i^j, \mathit{vk}_{\mathsf{CI}}, \{\mathit{id}, r\})\ \wedge \\ \quad \hat{p}_i \leftarrow \mathsf{E}(\mathit{ek}_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(\mathit{id}, \mathit{svc}))\ \wedge\ \hat{\mathit{tkn}}_i \leftarrow \mathsf{E}(\mathit{ek}_{\mathsf{SP}}, \mathit{tkn}_i)\ \wedge \\ \quad \hat{v}_i \leftarrow \mathsf{E}(\mathit{ek}_{\mathsf{SP}}, v)\ \wedge\ v \in C \end{array}\right\}$$

is realized as follows. The first two conjuncts have already been explained for P_j^1 above. The third conjunct is new: let σ_i^j be the signature for attribute at_j. Then we first randomize σ_i^j as $\tilde{\sigma}_i^j = ((\sigma_{i,1}^j)^t, (\sigma_{i,2}^j(\sigma_{i,1}^j)^s)^t)$ and start its verification by computing

$$\hat{\sigma}_i^j = e(\tilde{\sigma}_{i,2}^j, h) \cdot e(\tilde{\sigma}_{i,1}^j, X)^{-1} \cdot e(\tilde{\sigma}_{i,1}^j, Y_1)^{-\mathit{id}} \cdot e(\tilde{\sigma}_{i,1}^j, Y_2)^{-r} \cdot e(\tilde{\sigma}_{i,1}^j, h)^s. \tag{C.2}$$

We shall observe that $\hat{\sigma}_i^j = e(\tilde{\sigma}_{i,1}^j, Y_3)^{\mathit{at}_i^j} \cdot e(\tilde{\sigma}_{i,1}^j, Y_4)^{r_j}$ where r_j is encrypted in $(\bar{oi}_i)_j$. This latter part is, however, independent of the proof and is verified in a later stage of the Tally protocol. The proof just contains as a statement that the reduced signature is correctly computed from the known identity and the known values r and s (see (C.2)). The proof for the fourth conjunct of P_i has also been explained above. The proof for the fifth conjunct $\hat{\mathit{tkn}}_i \leftarrow \mathsf{E}(\mathit{ek}_{\mathsf{SP}}, \mathit{tkn}_i)$ is slightly tricky: we use a technique explained in [55], which does not really prove knowledge of tkn_i but only that the one who created the proof also created the encryption. This is sufficient for our purposes as well. We need the conjunct only to ensure that a malicious user cannot use a token from a different ballot as its own one. This can be effectively prevented by being required to show that the encryption is self-created. The proof shows that, given an ElGamal ciphertext $(g^r, Z^r \cdot m)$, one knows the randomness r in the first component with respect to g. The second component is not affected and targeted by the proof. Consequently, we do not make any statement about the second component. It is important that tokens, which are incorrectly encrypted, e.g., by using different random values in both components, are detected in later consecutive verification phases. This is what we guarantee through our Tally protocol in the second to last step: only a validly encrypted token will pass the token validity check, in which the token is compared via PET to the recomputed token based on $\hat{p}_i$. Finally, the encryption of the rating and checking that the rating is in a certain set C is done using exponential ElGamal encryption and an OR-proof [58]. The complete proof reads as follows where (β_1, β_2) is the blinded randomization of σ_{id}.

$$P_i = PK \left\{ \begin{array}{l} (\alpha, s, \rho, \mu_{[\ell]}, \zeta_{[4]}, \nu) : \\ \quad e(\beta_1, X_{\mathsf{PKI}}) \cdot e(\beta_1, Y_{\mathsf{PKI}})^{\alpha} \cdot (\beta_1, h)^{s} = e(\beta_2, h) \wedge \\ \quad e(\tilde{\sigma}^0_{i,1}, X_{\mathsf{CI}}) \cdot e(\tilde{\sigma}^0_{i,1}, Y_{1,\mathsf{CI}})^{\alpha} \cdot e(\tilde{\sigma}^0_{i,1}, Y_{2,\mathsf{CI}})^{\rho} \cdot (\tilde{\sigma}^0_{i,1}, h)^{\mu_0} = e(\beta_2, h) \wedge \\ \quad \forall 1 \leq j \leq \ell. \\ \qquad \hat{\sigma}^j_i = e(\tilde{\sigma}^j_{i,2}, h) \cdot e(\tilde{\sigma}^j_{i,1}, X_{\mathsf{CI}})^{-1} \cdot e(\tilde{\sigma}^j_{i,1}, Y_{1,\mathsf{CI}})^{-\alpha} \cdot \\ \qquad e(\tilde{\sigma}^j_{i,1}, Y_{2,\mathsf{CI}})^{-\rho} \cdot e(\tilde{\sigma}^j_{i,1}, h)^{\mu_j} \wedge \\ \quad \hat{p}_i = (g^{\zeta_1}, Z^{\zeta_1} \cdot \mathcal{S}^{\alpha}) \wedge \\ \quad t\hat{k}n_i = (t\hat{k}n_{i,1}, t\hat{k}n_{i,2}) = ((g^{\zeta_2}, t\hat{k}n_{i,1,2}), (g^{\zeta_3}, t\hat{k}n_{i,2,2})) \wedge \\ \quad \hat{v}_i = (g^{\zeta_4}, Z^{\zeta_4} \cdot g^{\nu}) \wedge \\ \quad \exists x \in C.\ x = g^{\nu} \end{array} \right\}$$

4. Finally, in the Tally protocol we require proofs of correct decryption for implementing the function DProto, plaintext-equivalence-tests, both of which are explained in detail in [55], and we need verifiable mixing networks (e.g., [110]) or even proofs of shuffle correctness [22] (depending on the required robustness, i.e., the number of malicious parties).

All these proofs can be made non-interactive by applying the Fiat-Shamir heuristic [78].

C.3. Proofs of Security and Privacy

Proof of Theorem 4.1. We define the following sequence of games, for which we will show that switching from one game to the next cannot be distinguished except with negligible probability. Since there are only polynomially many such game hops, the theorem follows.

- $\mathsf{Exp}_0^{\mathcal{A}}(\lambda) = \mathsf{Exp}_{\mathsf{AH}}^{\mathcal{A}}(\lambda, b)$

- $\mathsf{Exp}_{1,j}^{\mathcal{A}}(\lambda)$ is a series of $N+1$ games where $N = N_1(2m\ell+m)+N_2m+m+3N_3m+N_4m$, $\mathsf{Exp}_{1,0}^{\mathcal{A}}(\lambda) = \mathsf{Exp}_{0}^{\mathcal{A}}(\lambda)$, and $\mathsf{Exp}_{1,i}^{\mathcal{A}}(\lambda)$ is defined as $\mathsf{Exp}_{1,i-1}^{\mathcal{A}}(\lambda)$ with the difference that the i-th zero-knowledge proof in the Tally protocol is simulated whenever it is computed by an honest service provider.
- $\mathsf{Exp}_{2,i}^{\mathcal{A}}(\lambda)$ is a series of games where $\mathsf{Exp}_{2,0}^{\mathcal{A}}(\lambda) = \mathsf{Exp}_{1,N}^{\mathcal{A}}(\lambda)$ and $\mathsf{Exp}_{2,i}^{\mathcal{A}}(\lambda)$ is defined as $\mathsf{Exp}_{2,i-1}^{\mathcal{A}}(\lambda)$ with the following difference: if $i \notin I^*$, i.e., i is the index of an attribute that is different in both challenge sets of attributes, CI acts as follows in the challenge phase. CI does not encrypt the real opening information oi_i for SP but rather a different random value r_i.
- $\mathsf{Exp}_{3,i}^{\mathcal{A}}(\lambda)$ is a series of games where $\mathsf{Exp}_{3,0}^{\mathcal{A}}(\lambda) = \mathsf{Exp}_{2,\ell}^{\mathcal{A}}(\lambda)$ and $\mathsf{Exp}_{3,i}^{\mathcal{A}}(\lambda)$ is defined as $\mathsf{Exp}_{3,i-1}^{\mathcal{A}}(\lambda)$ with the following difference: if $i \notin I^*$, CI uses a random value s_i for at_i^b when forming $\sigma_{i^*}^i$.
- Obverse that $\mathsf{Exp}_{3,\ell}^{\mathcal{A}}(\lambda)$ is independent of b since all attributes that differ in both challenge vectors have been replaced by random values.

Claim: $\mathsf{Exp}_{0}^{\mathcal{A}}(\lambda) = \mathsf{Exp}_{1,0}^{\mathcal{A}}(\lambda)$, $\mathsf{Exp}_{1,N}^{\mathcal{A}}(\lambda) = \mathsf{Exp}_{2,0}^{\mathcal{A}}(\lambda)$, $\mathsf{Exp}_{2,\ell}^{\mathcal{A}}(\lambda) = \mathsf{Exp}_{3,0}^{\mathcal{A}}(\lambda)$. Hold by definition.

Claim: $\forall 1 \leq i \leq N.\ \mathsf{Exp}_{1,i}^{\mathcal{A}}(\lambda) \approx \mathsf{Exp}_{1,i-1}^{\mathcal{A}}(\lambda)$. Assume first of all that all proofs in the Tally oracle are expanded, e.g., all proofs which are a result of the DProto function are actually m sub-proofs, one for each service provider. Moreover, except for the verifiable mix network, which we consider as a proof as well, all remaining proofs are proofs for the same logical statement (of course with different public values and witnesses), namely:

$$PK\left\{(\alpha) :\ (ek, \alpha) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda) \wedge D = \mathsf{PartD}(\alpha, c)\right\} \tag{C.3}$$

for some ek, D, and c. The mixing proof can be formulated as

$$PK\left\{(\pi, \vec{r}_1, \vec{r}_2, \vec{r}_3) :\ \forall i.\ B_i' = \begin{pmatrix} \hat{p}_i' \\ tkn_i' \\ \hat{v}_i' \end{pmatrix} = \begin{pmatrix} \mathsf{Rnd}(ek_{\mathsf{SP}}, \hat{p}_{\pi^{-1}(i)}, (\vec{r}_1)_{\pi^{-1}(i)}) \\ \mathsf{Rnd}(ek_{\mathsf{SP}}, tkn_{\pi^{-1}(i)}, (\vec{r}_2)_{\pi^{-1}(i)}) \\ \mathsf{Rnd}(ek_{\mathsf{SP}}, \hat{v}_{\pi^{-1}(i)}(\vec{r}_3)_{\pi^{-1}(i)}) \end{pmatrix}\right\} \tag{C.4}$$

Hence, we distinguish cases in the following:

Case $i \leq N_1(2m\ell + m) + N_2m$ **or** $i > N_1(2m\ell + m) + N_2m + m$: In that case, we are confronted with replacing a proof of the form like in Equation (C.3). Let j be the index of the service provider, who is to be creating that proof. If $j \in Cor$, we simply do nothing and the claim follows since the game does not change. If, however, $j \notin Cor$, assume that there exists an adversary $\mathcal{A}$ that can distinguish $\mathsf{Exp}_{1,i}^{\mathcal{A}}(\lambda)$ and $\mathsf{Exp}_{1,i-1}^{\mathcal{A}}(\lambda)$ with non-negligible probability. Then we construct a reduction $\mathcal{B}$ against the zero-knowledge of ZKP.

Simulation: $\mathcal{B}$ bootstraps the system and simulates the interfaces as defined in the definition and the construction. When $\mathcal{A}$ calls the oracle $\mathcal{O}_{\mathsf{Tally}}$, $\mathcal{B}$ uses the simulator to generate proofs for the first $i-1$ proofs whenever the responsible service provider is not corrupted. For all proofs other proofs except the i-th one, it computes real proofs. Consider the i-th proof: let ek, D, c be the public values of the target proof and dk_j be the witness of SP_j. $\mathcal{B}$ selects a bit b and if $b=0$ it computes a real proof π^* while if $b=1$ it uses the simulator to construct π^*. It posts π^* on BB as the i-th proof. At some point, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs 1 if and only if $b=b'$.

Analysis: the above reduction is poly-time and perfectly simulates the games that $\mathcal{A}$ expects to see; in particular, if $b=0$, i.e., $\mathcal{B}$ creates a real proof, $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathcal{A}}_{1,i-1}(\lambda)$, otherwise, i.e., $\mathcal{B}$ uses the simulator to create the i-th proof, $\mathcal{B}$ simulates $\mathsf{Exp}^{\mathcal{A}}_{1,i}(\lambda)$. Additionally, $\mathcal{B}$ does not deviate in any protocol from its description. Hence, the changes performed by $\mathcal{B}$ are solely presented by the replaced zero-knowledge proof. Hence, $\mathcal{A}$'s advantage, which was assumed to be non-negligible, carries over one-to-one to that of $\mathcal{B}$, which means that $\mathcal{B}$ breaks zero-knowledge with non-negligible probability, a contradiction.

Case $i = N_1(2m\ell+m)+N_2m+j$ **for** $1 \le j \le m$: The proof is similar to the one above, just that the values used to create the proof consist of the public components $\vec{B}_j$, $\vec{B}_{j-1}$ and witnesses π_j, $\vec{r}_{1,j}$, $\vec{r}_{2,j}$, and $\vec{r}_{3,j}$. $\mathcal{B}$ then either creates a real proof or a simulated proof for the proof in Equation (C.4). The rest of the simulation and the analysis is equivalent to the previous proof.

Claim: $\forall 1 \le i \le \ell.\ \mathsf{Exp}^{\mathcal{A}}_{2,i}(\lambda) \approx \mathsf{Exp}^{\mathcal{A}}_{2,i-1}(\lambda)$. By definition, we have that $\mathsf{Exp}^{\mathcal{A}}_{2,i}(\lambda) = \mathsf{Exp}^{\mathcal{A}}_{2,i-1}(\lambda)$ whenever $i \in I^*$. Hence, the following reduction only regards the case where $i \notin I^*$. Assume towards contradiction that there exists an algorithm $\mathcal{A}$ that can distinguish $\mathsf{Exp}^{\mathcal{A}}_{2,i}(\lambda)$ from $\mathsf{Exp}^{\mathcal{A}}_{2,i-1}(\lambda)$ with non-negligible probability. We construct a reduction $\mathcal{B}$ against the CPA security of $\Pi_{\mathsf{PKE}}{}^{+}$.

Simulation: $\mathcal{B}$ gets a key tuple $(ek, dk_1, \ldots, dk_{m-1})$ from the $\Pi_{\mathsf{PKE}}{}^{+}$ challenger. It then runs the Setup protocol using those keys. The oracles in $\mathbb{O}$ are simulated as described in the construction and in the definition. No special care has to be taken for them.

Upon receiving the challenge $(i^*, \vec{at}_0, \vec{at}_1)$, $\mathcal{B}$ behaves as follows: it aborts if ClReg has already been called on i^* or if i^* is coerced. Otherwise, it generates the signatures $(\vec{\sigma}_b, \vec{aux}_b, \vec{oi}_b)$ for the attributes $\vec{at}_b$, as defined in the construction and prepares two challenges for the IND-CPA challenger: $\mathcal{B}$ sets $m_0 = (\vec{oi}_b)_i$ and $m_1 = r_i$ where r_i are chosen uniformly at random. $\mathcal{B}$ sends (m_0, m_1) to the IND-CPA challenger, who answers with c. Along with $(\sigma^1_b, aux^1_b), \ldots, (\sigma^\ell_b, aux^\ell_b)$, $\mathcal{B}$ sends $(c_1, \ldots, c_\ell)|_I$ to U_{i^*} where $c_j = \mathsf{E}^+(ek, r_j)$ for $j < i$ and $j \notin I^*$, and $c_j = \mathsf{E}^+(ek, (\vec{oi}_b)_j)$ for $j > i$. If there were more services, $\mathcal{B}$ would have to replace the encrypted opening information for each service provider one-by-one via hybrid argument by playing against different instances of the IND-CPA challenger.

We simulate the oracle $\mathcal{O}_{\mathsf{Tally}}$ as follows: since it aborts whenever $I \not\subseteq I^*$ and we only replace those opening information for values $i \notin I^*$, we have that all values $i \in I^*$ and hence all values $i \in I$ have still their correct opening information in place for verifying a signature. Consequently, since $\mathcal{B}$ stores this opening information during the simulation of

$\mathcal{O}_{\mathsf{ClReg}}$ and $\mathcal{O}_{\mathsf{Rate}}$ (in which the opening information is adapted), it can successfully simulate its decryption. Observe that the proofs of correct decryption are already simulated. For the ballots posted from information X using $\mathcal{O}_{\mathsf{PostBallot}}$, $\mathcal{B}$ simulates the decryption in the same way since X has to necessarily contain the encrypted opening information, which is either already known to $\mathcal{B}$, and some adapting opening information. Or it has to contain the randomness used to encrypt the given opening information, in case it differs from that generated in the beginning. Importantly, $\mathcal{B}$ has to reject all information X that does not allow him to access the opening information.

At some point, $\mathcal{A}$ outputs b^*, and $\mathcal{B}$ forwards that to the IND-CPA challenger.

Analysis: The above simulation is obviously poly-time and perfectly simulates $\mathsf{Exp}^{\mathcal{A}}_{2,i-1}(\lambda)$ whenever the IND-CPA challenger's bit is 0, and $\mathsf{Exp}^{\mathcal{A}}_{2,i}(\lambda)$ otherwise. Hence, since $\mathcal{A}$ can distinguish the two games with non-negligible advantage, so can $\mathcal{B}$ break the IND-CPA game with the same advantage. This is a contradiction to the IND-CPA security of $\Pi_{\mathsf{PKE}}{}^{+}$. The claim follows.

Claim: $\forall 1 \leq i \leq \ell$. $\mathsf{Exp}^{\mathcal{A}}_{3,i}(\lambda) \approx \mathsf{Exp}^{\mathcal{A}}_{3,i-1}(\lambda)$**.** By definition, we have that $\mathsf{Exp}^{\mathcal{A}}_{3,i}(\lambda) = \mathsf{Exp}^{\mathcal{A}}_{3,i-1}(\lambda)$ whenever $i \in I^*$. Hence, the following reduction only regards the case where $i \notin I^*$. Assume towards contradiction that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathsf{Exp}^{\mathcal{A}}_{3,i}(\lambda)$ and $\mathsf{Exp}^{\mathcal{A}}_{3,i-1}(\lambda)$ with non-negligible probability. Then we construct $\mathcal{B}$ against the hiding property of Π_{SRC}.

Simulation: $\mathcal{B}$ receives vk from the hiding challenger and runs the Setup protocol using vk, which it also forwards to $\mathcal{A}$. The oracles in $\mathbb{O}$ are simulated as described in the construction and the definition except for $\mathcal{O}^0_{\mathsf{ClReg}}(j, \vec{at}_j)$: $\mathcal{B}$ forwards a corresponding query for generating the SRC to the hiding challenger's signing oracle and uses henceforth whatever the oracle answers.

Upon receiving the challenge $(i^*, \vec{at}_0, \vec{at}_1)$, $\mathcal{B}$ checks whether ClReg has already been called on i^* or if i^* is coerced. If so, it aborts. Otherwise, it selects r for U_{i^*} and t_i uniformly at random. Then, it sends $((id_{i^*}, r, (\vec{at}_b)_i), (id_{i^*}, r, t_i))$ to the hiding challenger, who answers with (σ_j, aux_j). Furthermore, for all $j < i$ and $j \notin I^*$ it asks the oracle to sign-commit (id_{i^*}, r, t_j) for random values t_j while for all $j > i$ or $j < i$ and $j \in I^*$ it asks the oracle to sign-commit $(id_{i^*}, r, (\vec{at}_b)_j)$. Let the answers be $(\sigma_1, aux_1, oi_1), \ldots, (\sigma_{i-1}, aux_{i-1}, oi_{i-1}), (\sigma_{i+1}, aux_{i+1}, oi_{i+1}), \ldots, (\sigma_\ell, aux_\ell, oi_\ell)$. Finally, $\mathcal{B}$ prepares the answer for $\mathcal{A}$ as follows: for all $i \neq j \in I^*$, it encrypts oi_j for SP. For all $j \notin I^*$ (including i) it chooses random values r_j and encrypts those. The resulting vector $(\sigma_1, aux_1, \bar{oi}_1), \ldots, (\sigma_\ell, aux_\ell, \bar{oi}_\ell)$ is then sent to $\mathcal{A}$. The oracles that require signing are answered by using the corresponding oracles of the hiding challenger that are still provided after the challenge.

We notice that the oracle $\mathcal{O}_{\mathsf{Tally}}$ can be simulated trivially since all information related to indices in I is not changed. That is sufficient since only SRCs related to attributes in I as well as corresponding opening information is part of the ballots posted on BB and processed in Tally. So Tally is not affected by this reduction.

At some point, $\mathcal{A}$ outputs a bit b', which $\mathcal{B}$ forwards to the hiding challenger.

Analysis: We notice that the simulation is poly-time and correct. If the hiding challenger's bit is 0, we simulate $\mathsf{Exp}_{3,i-1}^{\mathcal{A}}(\lambda)$ and $\mathsf{Exp}_{3,i}^{\mathcal{A}}(\lambda)$ otherwise. Since $\mathcal{A}$ distinguishes $\mathsf{Exp}_{3,i-1}^{\mathcal{A}}(\lambda)$ and $\mathsf{Exp}_{3,i}^{\mathcal{A}}(\lambda)$ with non-negligible probability and $\mathcal{B}$ solely passes messages back and forth from the challenger to $\mathcal{A}$, $\mathcal{B}$ wins the hiding game with the same probability. This concludes the proof.

Conclusion. We have shown that $\mathsf{Exp}_{\mathsf{AH}}^{\mathcal{A}}(\lambda, b) = \mathsf{Exp}_{0}^{\mathcal{A}}(\lambda) \approx \mathsf{Exp}_{3,\ell}^{\mathcal{A}}(\lambda)$ and that $\mathsf{Exp}_{3,\ell}^{\mathcal{A}}(\lambda)$ is independent of b. Hence, any adversary cannot do better than guessing when being confronted with $\mathsf{Exp}_{e,\ell}^{\mathcal{A}}(\lambda)$, which concludes the proof. $\square$

Proof of Theorem 4.2. We prove the theorem by game hopping. In particular, we start with $\mathsf{Exp}_{\mathsf{Anon}}^{\mathcal{A}}(\lambda, 0)$ and gradually modify that game into $\mathsf{Exp}_{\mathsf{Anon}}^{\mathcal{A}}(\lambda, 1)$, while showing that each intermediate change can be distinguished by $\mathcal{A}$ only with negligible probability. The intermediate games are defined as follows:

- $\mathsf{Exp}_{0}^{\mathcal{A}}(\lambda, b) = \mathsf{Exp}_{\mathsf{Anon}}^{\mathcal{A}}(\lambda, b)$.
- $\mathsf{Exp}_{1,i}^{\mathcal{A}}(\lambda, b)$ is a series of $N + 1$ games where $N = N_1(2m\ell + m) + N_2 m + m + 3N_3 m + N_4 m$, $\mathsf{Exp}_{1,0}^{\mathcal{A}}(\lambda, b) = \mathsf{Exp}_{0}^{\mathcal{A}}(\lambda, b)$ and $\mathsf{Exp}_{1,i}^{\mathcal{A}}(\lambda, b)$ is defined as $\mathsf{Exp}_{1,i-1}^{\mathcal{A}}(\lambda, b)$ with the difference that the i-th zero-knowledge proof output in the Tally oracle, when generated by an honest service provider, is now simulated.
- $\mathsf{Exp}_{2}^{\mathcal{A}}(\lambda, b)$ is defined as $\mathsf{Exp}_{1,N}^{\mathcal{A}}(\lambda, b)$ with the difference that the proof P_{i_b} submitted by the challenge user U_{i_b} in the ballot is simulated now.
- $\mathsf{Exp}_{3,i}^{\mathcal{A}}(\lambda, b)$ is a series of $n + 1$ games where $\mathsf{Exp}_{3,0}^{\mathcal{A}}(\lambda, b) = \mathsf{Exp}_{2}^{\mathcal{A}}(\lambda, b)$ and $\mathsf{Exp}_{3,i}^{\mathcal{A}}(\lambda, b)$ is defined as $\mathsf{Exp}_{3,i-1}^{\mathcal{A}}(\lambda, b)$ with the difference that when $\mathcal{O}_{\mathsf{SPReg}}$ is called on i, then the proofs P_j^1 are simulated. This requires another hybrid argument since there are m such proofs.
- $\mathsf{Exp}_{4,j}^{\mathcal{A}}(\lambda, b)$ is a series of $m+1$ games where $\mathsf{Exp}_{4,0}^{\mathcal{A}}(\lambda, b) = \mathsf{Exp}_{3,n}^{\mathcal{A}}(\lambda, b)$ and $\mathsf{Exp}_{4,j}^{\mathcal{A}}(\lambda, b)$ is defined as $\mathsf{Exp}_{4,j-1}^{\mathcal{A}}(\lambda, b)$ with the difference that the j-th designated verifier proof P_j^2 in the oracle $\mathcal{O}_{\mathsf{SPReg}}$, called on any i, when generated by an honest service provider, is now simulated. Since there are n such proofs to be replaced in each game hop, we need another hybrid argument to replace all n (at most) proofs one after another.
- The only difference between $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, 0)$ and $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, 1)$ is that instead of the encrypted pseudonym and token for user U_{i_0}, the ballot that is posted in the challenge phase on BB now contains those information for U_{i_1}.

Before we start proving that distinguishing all of these games is hard, we argue why we can ignore the oracle $\mathsf{PostBallot}$ in all following reductions. Using that oracle, the adversary can post arbitrary messages on BB, in particular self-crafted ballots (everything secret information necessary to generate a ballot such as the randomness used for the proof or for the encryption has to pass through the challenger). We notice that the adversary may not coerce the users U_{i_0} and U_{i_1}, hence, he may not get any secret information of these parties. Consequently, even though one of those two users posts a ballot on BB, the adversary does have nothing at disposal with which it could craft a ballot that he might link to the challenged ballot. More concretely, investigating the form of a ballot $B = ((\{\hat{\sigma}^j, \hat{oi}^j, \hat{at}^j\}_{j\in I}, \hat{p}, \hat{v}, \hat{tkn}, P)$, we notice that the only information, which is unique for each user, is the opening information encrypted in $\hat{oi}^j$, the user's encrypted pseudonym $\hat{p}$, and the user's token, which is encrypted in $\hat{tkn}$. The encrypted opening information is a random value, independent of any knowledge of $\mathcal{A}$ since an honest user chooses it uniformly random before preparing a ballot, hence no adversary can try to link two ballots based on that information. Since the other two parts, the pseudonym and token appear both as random values to $\mathcal{A}$ for uncoerced users, it can also not use them to form new ballots since the proof P would not verify, due to the lack of knowing the private identity of that respective user.

Claim: $\mathsf{Exp}_0^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{1,0}^{\mathcal{A}}(\lambda, b)$**,** $\mathsf{Exp}_2^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{3,0}^{\mathcal{A}}(\lambda, b)$**, and** $\mathsf{Exp}_{3,n}^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{4,0}^{\mathcal{A}}(\lambda, b)$**.** Hold by definition.

Claim: $\forall 1 \le i \le N.\ \mathsf{Exp}_{1,i}^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{1,i-1}^{\mathcal{A}}(\lambda, b)$**.** The proof is identical to the proof of the respective claim in the proof of Theorem 4.1.

Claim: $\mathsf{Exp}_{1,N}^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_2^{\mathcal{A}}(\lambda, b)$**,** $\forall 1 \le i \le n.\ \mathsf{Exp}_{3,i}^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{3,i-1}^{\mathcal{A}}(\lambda, b)$**, and** $\forall 1 \le j \le m.\ \mathsf{Exp}_{4,j}^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{4,j-1}^{\mathcal{A}}(\lambda, b)$**.** It is easy to see that the claims follow from the zero-knowledge of ZKP. The proofs are similar to the previous ones using the corresponding statements as well as public components and witnesses. The construction of the reduction as well as the analysis of their success probability is the same.

Claim: $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, 0) \approx \mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, 1)$**.** Assume that there exists an adversary $\mathcal{A}$ which is able to distinguish $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, 0)$ and $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, 1)$ with non-negligible advantage. We observe that all proofs generated by honest and not corrupted parties are simulated at this point in time. Moreover, the only information depending on the bit b available to $\mathcal{A}$ is the ballot submitted by the challenge user. This ballot has the form $(\{\hat{\sigma}_{i_b}^j, \hat{oi}_{i_b}^j, \hat{at}_{i_b}^j\}_{j\in I}, \hat{p}_{i_b}, \hat{v}_{i_b}, \hat{tkn}_{i_b}, P_{i_b})$. The first set consists of the reduced credentials and the encrypted opening information and attributes. The reduced credentials only contain the same attributes, independent of b (that was a requirement for the challenge user indices). Hence, also the attributes are the same independent of b. Finally, even though $\mathcal{A}$ generated the credentials and hence knew the original opening information, $\mathcal{B}$, who crafted the ballot, adapted the opening information in the course of preparing the ballot, using uniformly distributed randomness. Hence, also the opening information is independent of b. The last component of the ballot, the proof P_{i_b} is independent of b since it is simulated. The encrypted rating in $\hat{v}_{i_b}$ is independent of b since it is chosen the same, no matter for which value of b. Hence, we are left with $\hat{p}_{i_b}$ and $\hat{tkn}_{i_b}$, which are dependent on b. Since $\mathcal{A}$ can distinguish both games with non-negligible probability, it must be the case that it outputs the correct bit b with non-negligible probability. Since $\hat{p}_{i_b}$ and $\hat{tkn}_{i_b}$ are the only information on which it can base that decision (remember that all proofs that might have contained information are simulated), we investigate on these values further.

Let us analyze which additional information $\mathcal{A}$ has about the two challenge users U_{i_0} and U_{i_1}, which might never be coerced during the course of the game. It gathers information in $\mathcal{O}_{\mathsf{CIReg}}$, namely, the blinded identity bid_{i_0} and bid_{i_1} of either user. Moreover, in $\mathcal{O}_{\mathsf{SPReg}}$ it gathers the same information for all corrupted service providers. Furthermore, for $j \in Cor$, it receives $\mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(id_{i_0}, svc)^{s_{j,0}}$ and $\mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(id_{i_1}, svc)^{s_{j,1}}$ for two unknown random values $s_{j,0}, s_{j,1}$. So even if $\mathcal{A}$ corrupts all service providers and can hence decrypt the ciphertexts, obtaining $\mathsf{SSP}_{\mathbb{G}}(id_{i_0}, svc)^{s_{j,0}}$ and $\mathsf{SSP}_{\mathbb{G}}(id_{i_1}, svc)^{s_{j,1}}$, the decrypted value is a uniformly distributed value in $\mathbb{G}$, unlinkable to any of bid_{i_0} or bid_{i_1}. Consequently, the only information gathered by $\mathcal{A}$ during the registration oracles is the blinded identity bid_{i_0} and bid_{i_1} of either challenge user.

From the two values $\hat{p}_{i_b}$ and $\widehat{tkn}_{i_b}$, $\mathcal{A}$ (if it corrupted all service providers) can hence extract $psd = \mathsf{SSP}_{\mathbb{G}}(id_{i_b}, svc)$ and $bpsd = \mathsf{SSP}_{\mathbb{G}}(id_{i_b}, svc)^{\sum_{j=1}^{m} b_j}$. For $bpsd$, $\mathcal{A}$ can even remove the sum of the b_j since those are known to the service providers, too.

To summarize, $\mathcal{A}$ has gathered bid_{i_0}, bid_{i_1}, and psd (since $bpsd$ does not give any additional information with respect to psd). These three values are exactly an instance of the definition of anonymity for service-specific pseudonyms, in which the adversary is supposed to tell to which blinded identity a given pseudonym belongs. The formal reduction works as follows.

Simulation: $\mathcal{B}$ receives two blinded identities bid_0 and bid_1, which it assigns to two guessed users $\mathsf{U}_{i_0^*}$ and $\mathsf{U}_{i_1^*}$, respectively. Additionally, it receives a pseudonym psd^*. It uses psd^* for both users when $\mathcal{O}_{\mathsf{SPReg}}$ is called on any of them. Recall that all proofs generated by users are simulated, so the real identity is never needed to simulate any oracle except when preparing the ballot for the challenge user so as to generate the reduced signature. Hence, $\mathcal{B}$ will generate the same identification token for both $\mathsf{U}_{i_1^*}$ and $\mathsf{U}_{i_0^*}$.

When challenged, $\mathcal{B}$ aborts whenever $\{i_0, i_1\} \neq \{i_0^*, i_1^*\}$. Otherwise assume without loss of generality that $(i_0, i_1) = (i_0^*, i_1^*)$. $\mathcal{B}$ generates a reduced signature for any of the challenge users, no matter which one. The proof P in the ballot is simulated so the identity behind the proof does not matter, the only thing that matters is that the signature is correctly reduced and can be validated using the encrypted opening information and attributes and the presence of correct and validating values of $\hat{p}$ and $\widehat{tkn}$. In order to generate that reduced signature, $\mathcal{B}$ needs to know either $id_{i_0^*}$ or $id_{i_1^*}$, or the discrete logarithms of $\sigma^j_{i_0^*,1}$ or $\sigma^j_{i_1^*,1}$ for $j \in I \cup \{0\}$. Since the former is impossible, we can only rely on the latter possibility. These discrete logarithms can be extracted by using the knowledge extractor of the zero-knowledge proof of knowledge π sent by $\mathcal{A}$ in the $\mathcal{O}_{\mathsf{CIReg}}$ protocol. We call the extracted values $\alpha_0^0, \alpha_1^0, \dots, \alpha_\ell^0$ and $\alpha_0^1, \alpha_1^1, \dots, \alpha_\ell^1$, respectively. We exemplify the reduction of a single signature, the rest is analogous. Let $\sigma^j_{i_0} = (\sigma^j_{i_0,1}, \sigma^j_{i_0,2})$ be the j-th SRC of user U_{i_0}. Then we compute

$$\hat{\sigma}^j_{i_0} = e(\sigma^j_{i_0,2}, h) \cdot e(\sigma^j_{i_0,1}, X_{\mathsf{CI}}^{-1} \cdot Y_{2,\mathsf{CI}}^{-r}) \cdot \underbrace{e(bid_0, Y_{1,\mathsf{CI}}^{-\alpha_0^j})}_{=e(\sigma^j_{i_0,1}, Y_{1,\mathsf{CI}}^{id_0})},$$

obtaining a correctly reduced SRC. The rest of the simulation of the $\mathcal{O}_{\mathsf{Rate}}$ oracle works as before, just that $\hat{p} = \mathsf{E}(ek_{\mathsf{SP}}, psd^*)$ and $\widehat{tkn}$ is the encryption of the token of any of the two users (remember that they are the same). The remaining oracle is $\mathcal{O}_{\mathsf{Tally}}$, which is also simulated as before for honest service providers without any change, there is no difference to the previous game. Finally, $\mathcal{A}$ outputs a bit, which $\mathcal{B}$ forwards to the anonymity challenger of SSP.

Analysis: The simulation is correct and efficient, as we have demonstrated. Moreover, whenever the challenger's bit is 0, $\mathcal{B}$ effectively simulates $\mathsf{Exp}^{\mathcal{A}}_{4,m}(\lambda, 0)$ while it simulates $\mathsf{Exp}^{\mathcal{A}}_{4,m}(\lambda, 1)$ otherwise. Since $\mathcal{A}$ wins with non-negligible probability, and $\mathcal{B}$ forwards the bit given by $\mathcal{A}$ to the anonymity challenger, $\mathcal{A}$'s success probability carries over to $\mathcal{B}$, only lowered by a factor of $2/(n^2 - n)$ due to the guessing of the correct challenge users.

Conclusion. In total, we have shown that $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Anon}}(\lambda, 0) = \mathsf{Exp}^{\mathcal{A}}_{0}(\lambda, 0) \approx \mathsf{Exp}^{\mathcal{A}}_{0}(\lambda, 1) = \mathsf{Exp}^{\mathcal{A}}_{\mathsf{Anon}}(\lambda, 1)$, which concludes the proof for the theorem. □

Proof of Theorem 4.3. We prove the theorem by game hopping. In particular, we start with $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{CR}}(\lambda, 0)$ and gradually modify that game into $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{CR}}(\lambda, 1)$, while showing that each intermediate change can be distinguished by $\mathcal{A}$ only with negligible probability. The intermediate games are defined as follows:

- $\mathsf{Exp}^{\mathcal{A}}_{0}(\lambda, b) = \mathsf{Exp}^{\mathcal{A}}_{\mathsf{CR}}(\lambda, b)$.

- $\mathsf{Exp}^{\mathcal{A}}_{1,i}(\lambda, b)$ is a series of $N + 1$ games where $N = N_1(2m\ell + m) + N_2 m + m + 3N_3 m + N_4 m$, $\mathsf{Exp}^{\mathcal{A}}_{1,0}(\lambda, b) = \mathsf{Exp}^{\mathcal{A}}_{0}(\lambda, b)$ and $\mathsf{Exp}^{\mathcal{A}}_{1,i}(\lambda, b)$ is defined as $\mathsf{Exp}^{\mathcal{A}}_{1,i-1}(\lambda, b)$ with the difference that the i-th zero-knowledge proof output in the Tally oracle, when generated by an honest service provider, is now simulated.

- $\mathsf{Exp}^{\mathcal{A}}_{2}(\lambda, b)$ is defined as $\mathsf{Exp}^{\mathcal{A}}_{1,N}(\lambda, b)$ with the difference that the proofs P_j and P_k submitted by the balancing users U_j and U_k in their respective ballots are simulated now.

- $\mathsf{Exp}^{\mathcal{A}}_{3,i}(\lambda, b)$ is a series of $n + 1$ games where $\mathsf{Exp}^{\mathcal{A}}_{3,0}(\lambda, b) = \mathsf{Exp}^{\mathcal{A}}_{2}(\lambda, b)$ and $\mathsf{Exp}^{\mathcal{A}}_{3,i}(\lambda, b)$ is defined as $\mathsf{Exp}^{\mathcal{A}}_{3,i-1}(\lambda, b)$ with the difference that when $\mathcal{O}_{\mathsf{SPReg}}$ is called on i, then the proofs P^1_j are simulated. This requires another hybrid argument since there are m such proofs.

- $\mathsf{Exp}^{\mathcal{A}}_{4,j}(\lambda, b)$ is a series of $m+1$ games where $\mathsf{Exp}^{\mathcal{A}}_{4,0}(\lambda, b) = \mathsf{Exp}^{\mathcal{A}}_{3,n}(\lambda, b)$ and $\mathsf{Exp}^{\mathcal{A}}_{4,j}(\lambda, b)$ is defined as $\mathsf{Exp}^{\mathcal{A}}_{4,j-1}(\lambda, b)$ with the difference that the j-th designated verifier proof P^2_j in the oracle $\mathcal{O}_{\mathsf{SPReg}}$, called on any i, when generated by an honest service provider, is now simulated. Since there are n such proofs to be replaced in each game hop, we need another hybrid argument to replace all n (at most) proofs one after another.

- $\mathsf{Exp}^{\mathcal{A}}_{5}(\lambda, b)$ is defined as $\mathsf{Exp}^{\mathcal{A}}_{4,m}(\lambda, b)$ with the difference that the decrypted blinded pseudonyms of all users in step 2 (duplicate elimination) of the Tally protocol are replaced with uniformly random values.

- The only difference between $\mathsf{Exp}^{\mathcal{A}}_{5}(\lambda, 0)$ and $\mathsf{Exp}^{\mathcal{A}}_{5}(\lambda, 1)$ is that the pseudonyms and tokens are replaced for users U_{i^*}, U_j, and U_k, according to the definition.

Claim: $\mathsf{Exp}_0^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, b)$**.** The proofs for those game hops are analogous to the proofs of the same claims in the proof of Theorem 4.2. The only difference is that there is always at least one not corrupted service provider for which proofs have to simulated.

Claim: $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, b) \approx \mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, b)$**.** Assume that there exists an adversary $\mathcal{A}$ who is able to distinguish $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, b)$ and $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, b)$ with non-negligible probability. We construct an adversary $\mathcal{B}$ against the hardness of DDH.

Simulation: The DDH challenger outputs a DDH tuple (h, h^x, h^y, h^z) and $\mathcal{A}$ outputs (I, n, m). $\mathcal{B}$ chooses an index j uniformly at random in $\{1, \dots, m\}$ as a guess for a service provider that will not be corrupted during the game. When bootstrapping the system, $\mathcal{B}$ chooses $s \in \mathbb{Z}_q^*$ and announces $svc = h^s$. It sets $g = h^x$, $B_i = g^{b_i}$ for random $b_i \in \mathbb{Z}_q^*$ for all $i \neq j$ and $B_j = h^z$. The simulation of the oracles starts here and happens as described. When $\mathcal{A}$ calls $\mathsf{Corrupt}(j)$, $\mathcal{B}$ aborts the simulation. We have to take special care of the oracles $\mathcal{O}_{\mathsf{SPReg}}$ and $\mathcal{O}_{\mathsf{Tally}}$ since they require knowledge of the, to SP_j, unknown b_j which is exactly zx^{-1}.

We start with $\mathcal{O}_{\mathsf{SPReg}}$. Recall that the entire interaction of $\mathcal{O}_{\mathsf{SPReg}}$ between the user and SP_j happens locally on $\mathcal{B}$'s side since SP_j cannot be corrupted. Hence, when $\mathcal{A}$ calls $\mathcal{O}_{\mathsf{SPReg}}(i)$, $\mathcal{B}$ selects for each $k \neq j$ s_k at random and computes $tkn_k \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}(id_i, svc)) = \mathsf{E}(ek_{\mathsf{SP}}, svc^{id_i})^{s_k}$, which it sends to each service provider. When it is about to interact with SP_j, $\mathcal{B}$ computes the following: it does not know the randomness to blind tkn_j but it knows the service descriptions exponent. Notice that the blinding step using exponentiation to s_j can be left out since SP_j cannot be corrupted. Hence, it now computes $tkn_i^j \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, (h^y)^{id_i \cdot s})$.

The very same computation is carried out when $\mathcal{B}$ plays SP_j in the $\mathcal{O}_{\mathsf{Tally}}$ in step 4, where the encrypted pseudonyms are blinded by all SP's before comparing them to the tokens. We need to treat one subtle difficulty in more detail though. In step 4 of the Tally protocol, the knowledge of which pseudonym is contained in which ballot is gone. Hence, $\mathcal{B}$ has to decrypt $\hat{p}_i$ for every ballot and check by recomputation of the SSP which signing key was used to generate the contained pseudonym. This is possible due to the honest setup of the service providers, hence $\mathcal{B}$ knows all decryption keys. If no fitting signing key is known, it means that $\mathcal{A}$ must have injected a ballot with an incorrect identity via $\mathsf{PostBallot}$. Then, $\mathcal{B}$ has to use the knowledge extractor of the zero-knowledge proof of knowledge system on the original ballot list to extract all identities from the ballots' proof P. Let the fitting identity be id'. Starting from here on, the computation is the same as described above. Notice that we do not have to consider any correctness proofs since they are simulated. The rest of the simulation is as described in the definition and according to the protocol. Finally, $\mathcal{B}$ forwards the bit output by $\mathcal{A}$.

Analysis: The simulation is efficient since the number of users in the system and hence, the number of identities as well as the number of service providers is polynomially bounded. We have to show that the simulation is correct. Let $z = xy$, i.e., the DDH tuple is valid. Let us consider the blinded pseudonyms rt_i for $1 \leq i \leq \ell$ that are decrypted in step 4 of the Tally protocol:

$$rt_i = svc^{id \cdot (b_1 + \cdots + b_{j-1} + y + b_{j+1} + \cdots + b_m)}$$

while $B_j = h^y$, i.e., rt_i is related to B_j and the relation is known. One can decompose rt_i into the separate parts $svc^{id \cdot b_k}$ and $svc^{id \cdot y}$ respectively, which are all related to their counterparts B_k and B_j via the equality of the discrete logarithm. Moreover, these are exactly the conditions to be met for $\mathcal{B}$ perfectly simulating $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, b)$. Now let z be a uniformly random value. Then B_j is a uniformly random value with respect to rt_i, i.e., the relation as in the previous case does no longer hold. Hence, rt_i is also a uniformly random value with respect to B_j, which means that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, b)$. As we assumed that $\mathcal{A}$ can distinguish $\mathsf{Exp}_{4,m}^{\mathcal{A}}(\lambda, b)$ and $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, b)$ with non-negligible advantage, $\mathcal{B}$ has almost the same advantage, only lowered by the correct guess of j, which lowers the advantage by a factor of $1/m$. This is still non-negligible since $m \in \mathsf{poly}(\lambda)$ and hence, a contradiction to the hardness of DDH. The claim follows.

Claim: $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, 0) \approx \mathsf{Exp}_5^{\mathcal{A}}(\lambda, 1)$. Assume that there exists an adversary $\mathcal{A}$ which is able to distinguish $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, 0)$ and $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, 1)$ with non-negligible probability. Then we construct an adversary $\mathcal{B}$ against the CPA-security of Π_{PKE}. We are using the hybrid version of CPA security, i.e., the challenger encrypts multiple messages to challenge ciphertexts.

Simulation: On input (I, n, m) by $\mathcal{A}$, $\mathcal{B}$ forwards m to the CPA challenger and receives in return $(ek, dk_1, \ldots, dk_{m-1})$, i.e., one partial decryption key is unknown. The bootstrapping phase proceeds as defined. Additionally, $\mathcal{B}$ guesses i^*, j, k, the users that $\mathcal{A}$ later on challenges on, and the two balancing users. When $\mathcal{A}$ calls $\mathcal{O}_{\mathsf{Corrupt}}(i)$, the challenger hands out one of the decryption keys and a random factor b_i where $B_i = g^{b_i}$. If $\mathcal{A}$ calls the interface more than $m-1$ times on different indices, $\mathcal{B}$ aborts the simulation. When $\mathcal{A}$ attempts to coerce any of i^*, j, or k, $\mathcal{B}$ aborts. The same happens when any of those are not registered with the same attribute sets (restricted to attributes in I). Before starting the simulation of $\mathcal{O}_{\mathsf{SPReg}}$, $\mathcal{B}$ prepares two challenge vectors for the CPA challenger, namely, $((\mathsf{SSP}(sk_{i^*}, svc), \mathsf{SSP}(sk_j, svc), r, \mathsf{SSP}(sk_j, svc), \mathsf{SSP}(sk_k, svc)), (s, \mathsf{SSP}(sk_k, svc), \mathsf{SSP}(sk_{i^*}, svc), \mathsf{SSP}(sk_k, svc), \mathsf{SSP}(sk_{i^*}, svc)))$ for random s and r, which stand for the faked tokens that we require for balancing the outcome in Tally. $\mathcal{B}$ sends those two vectors to the CPA challenger and receives $(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ in return. Whenever $\mathcal{O}_{\mathsf{SPReg}}$ is called on either i^*, j, or k, $\mathcal{B}$ uses either c_1^*, c_2^*, or c_3^* rather than encrypting the pseudonyms freshly. The values c_4^* and c_5^* are required later in the simulation of the challenge phase. The other oracles are simulated as defined in the definition and in the construction.

When $\mathcal{A}$ challenges $\mathcal{B}$ on i^*, $\mathcal{B}$ performs all checks outlined in the definition. If these checks are all passed and j and k are fitting choices for the balancing users, $\mathcal{B}$ starts preparing the ballots for U_j and U_k (U_k and U_{i^*}, respectively, which will be determined by the internal choice of the CPA challenger). It prepares the ballot for the first balancing user (which is either U_j or U_k) using $\hat{p}_1 = c_4^*$ and the encrypted token stored for user U_j. It prepares the ballot for the second balancing user (which is either U_k or U_{i^*}) using $\hat{p}_2 = c_5^*$ and the encrypted token stored for user U_k. It does not matter which SRC is used to create the ballot and which rating is encrypted since the proofs P in the ballots are simulated. Moreover, we will take care of creating the right result in the Tally protocol by taking the fixed constant result (not the proof, which is simulated) of the PET, which has been established in the course of simulating the correctness proof. The so created ballots are appended to BB, where $\mathcal{A}$ already placed its ballot for U_{i^*}.

In the simulation of the finalizing $\mathcal{O}_{\mathsf{Tally}}$, we have to make sure that any decryption is simulated correctly since $\mathcal{B}$ can, in fact, not decrypt due to the missing final decryption key. We show step-by-step how to simulate the decryptions. As a general remark, we notice that in order to simulate decryption correctly, it is sufficient to know the underlying plaintext since the required correctness proof for knowing the final (missing) decryption key is simulated, thus any plaintext could be used to simulate a correct decryption. The only crucial point to ensure is that the correct plaintext is used. Otherwise the Tally protocol might fail at some point or $\mathcal{A}$ might see that the simulation is not correct, given the information that $\mathcal{A}$ has at disposal.

- The first one is for a different encryption scheme, namely, $\Pi_{\mathsf{PKE}}{}^{+}$, and is thus not affected by the reduction. $\mathcal{B}$ can decrypt it without problem.
- The second decryption is for the attributes. Those are known to $\mathcal{B}$ due to the correctness proofs of $\mathcal{A}$ sent in $\mathcal{O}_{\mathsf{ClReg}}$.
- The third decryption is for the encrypted tokens, all of which $\mathcal{B}$ has generated. For ballots posted via $\mathcal{O}_{\mathsf{PostBallot}}$, $\mathcal{B}$ needs to use the extractor of the zero-knowledge proof system to extract the corresponding token from the proof P. This allows even to retrieve tokens that might have been randomized by $\mathcal{A}$ even though they had been generated by $\mathcal{B}$ in the first place or those that are malformed.
- The fourth decryption decrypts the token itself. The result of that decryption has been replaced in the previous game hop by random values. Hence, $\mathcal{B}$ chooses random values that are consistent with the identities behind the ballots so as to preserve linkability. We observe here that $\mathcal{B}$ also knows the plaintexts of all values contained in ballots created from X, which is submitted by $\mathcal{A}$ via $\mathcal{O}_{\mathsf{PostBallot}}$. So our simulation is not affected by malformed ballots or made up ballots submitted via $\mathcal{O}_{\mathsf{PostBallot}}$, since $\mathcal{A}$ always has to send all information necessary to open a ballot, independent of the decryption keys, in particular.
- The fifth decryption is for the PET. Here, $\mathcal{B}$ first tracks the permutations applied in the mixing networks (by using the zero-knowledge extractor). It then assigns the correct results to the PET. It can do that since it knows those results for every token on the board, including those submitted via $\mathcal{O}_{\mathsf{PostBallot}}$ (remember that $\mathcal{A}$ has to send either information that $\mathcal{B}$ already knows, such as a token generated during an honest call to $\mathcal{O}_{\mathsf{SPReg}}$, or it has to send all the information to access all plaintexts contained in a ballot, such as the randomness used for encrypting a message). Furthermore, it knows where the ballots moved during the execution of the mixing network due to the extracted permutations. The only ballots that are treated differently are those three submitted for i^* by $\mathcal{A}$ and for the two balancing users by $\mathcal{B}$. Here, two arbitrary ones are deemed to fulfill the PET while the remaining one is marked as invalid. Since the single permutation applied by the honest service provider does not allow $\mathcal{A}$ to link the results to the original ballots, this simulation is fine.

- The sixth decryption is for the ratings, which are extracted from the original proofs P accompanying every ballot using the extractor. Clearly, $\mathcal{B}$ has to track where each ballot went through the mixing network (it can just use those extracted values from the simulation of the PET). For the three ballots submitted by $\mathcal{A}$ for i^* and the balancing users by $\mathcal{B}$, one of which is now invalid, $\mathcal{B}$ simply chooses values v (which it has extracted on the initial bulletin board from the last ballot submitted for i^*) and an arbitrary rating v', those $\mathcal{A}$ expects to see.

Analysis: The simulation is efficient. We observe that in case the internal bit of the CPA challenger is 0, $\mathcal{B}$ perfectly simulates $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, 0)$ since the tokens used in the Rate protocol are those of U_{i^*}, given to $\mathcal{A}$, and those of U_j and a fake one. Furthermore, the ballots submitted for j and k contain encrypted pseudonyms for U_j, and U_k. Likewise, if that bit is 1, $\mathcal{B}$ perfectly simulates $\mathsf{Exp}_5^{\mathcal{A}}(\lambda, 1)$ since the tokens used in Rate are a fake one, given to $\mathcal{A}$, and those of U_k and U_{i^*} (who cheated on $\mathcal{A}$ by giving a fake token). Similarly, the encrypted pseudonyms used for the balancing ballots are those of U_k and U_{i^*}, coinciding with the valid tokens generated before. Since we assumed that $\mathcal{A}$ wins with non-negligible advantage and $\mathcal{B}$ does only use the information received by the CPA challenger in an unmodified manner, the advantage carries over to $\mathcal{B}$, only lowered by a factor of $3/(n(n-1)(n-2))$ for guessing the three user indices correctly. This is a contradiction to the CPA security of Π_{PKE}. The claim follows.

Conclusion. In total, we have shown that $\mathsf{Exp}_{\mathsf{CR}}^{\mathcal{A}}(\lambda, 0) = \mathsf{Exp}_0^{\mathcal{A}}(\lambda, 0) \approx \mathsf{Exp}_0^{\mathcal{A}}(\lambda, 1) = \mathsf{Exp}_{\mathsf{CR}}^{\mathcal{A}}(\lambda, 1)$, which concludes the proof for the theorem. $\square$

Proof of Theorem 4.4. The proof for recorded-as-cast individual verifiability is simple. The bulletin board is public and every user can see whether her ballot is contained in it. If that ballot is considered further in the duplicate elimination step in which RV is applied to the bulletin board, then that user can be sure that it will be considered in the final tally, unless it was a duplicate. A user can be assured that at least one of her submitted ballots ends up in the final tally because she knows whether her token is valid or not, and the validity has been proven by the service providers during the protocol SPReg. Of course, a coercer only knows that his ballot is considered in the tallying process but does not know whether his vote is considered in the final tally due to the fact that he cannot distinguish between a real and a fake token, albeit the existence of the proof transcripts from the SPReg protocol.

Albeit not achieving cast-as-intended individual verifiability by construction, CR^{2}P can be extended to that as follows using a technique by Adida [1]: assume that the user submits its rating via some rating software running on its device. The user does not necessarily trust that device (otherwise cast-as-intended individual verifiability would hold by trust). Hence, whenever the software outputs a ballot, the user has two choices: it can either use the ballot and submit it to the bulletin board or it can probe the software and ask for all random values used in the generation of the ballot so as to recompute the ballot and check whether all information that the user wanted to be included is included correctly in fact. This probing can be done an arbitrary number of times. It is straightforward to extend CR^{2}P with that technique by just showing all random values used in the Rate protocol.

Concerning clash-attacks, the argument is even simpler than the one for recorded-as-cast individual verifiability. Since every user submits her ballot on their own, they know the exact form of the ballot and can simply check the existence thereof on BB. Due to the randomized process of ballot creation and the randomness space (exponential in the security parameter), the probability that two ballots of two different users are equivalent, is negligible. Hence, clash-attacks do not exist in our protocol.

For the proof of universal verifiability, assume that there exists an adversary $\mathcal{A}$ which wins the universal verifiability game with non-negligible advantage, i.e., $\mathcal{A}$ is able to compute a tally V and a correctness proof π such that π verifies as being a proof of correctness for V given the initial bulletin board $\mathsf{BB} = (B_1, \dots, B_n)$. Moreover, we define V' as in the definition as the list of all ratings and user indices (v, j) for which there exists a ballot B_i with $\mathsf{Valid}_{\mathsf{BB}}^{\mathsf{RV}}(B_i, v, j)$. We recall the definition of $\mathsf{Valid}_{\mathsf{BB}}^{\mathsf{RV}}(B, v, j)$: $B = (\{\hat{\sigma}^j, \hat{oi}^j, \hat{at}^j\}_{j \in I}, \hat{p}, \hat{v}, \hat{tkn}, P)$ satisfies $\mathsf{Valid}_{\mathsf{BB}}^{\mathsf{RV}}(B, v, j)$ if

- P is valid with respect to B,
- the finishing signature verification succeeds on $\hat{\sigma}^j$ and the plaintexts of $\hat{oi}^j$ and $\hat{at}^j$, respectively,
- v is the rating encrypted in $\hat{v}$,
- and finally, $\hat{tkn}$ is the encryption of U_j's token.

Towards contradiction, we assume that there exists an adversary $\mathcal{A}$ which wins the verifiability game with non-negligible probability. Hence, it must be the case that $V \neq V''$ where V'' contains only the ratings extracted from the pairs in V' after duplicate elimination and the proof π verifies on BB. Consequently, there must either exist a ballot B_i in BB which is judged valid even though it is not or the proof π verifies even though it should not. We split $\mathcal{A}$'s success probability with respect to these two exhaustive events:

$$\begin{aligned}\mathsf{Pr}[\mathcal{A} \text{ wins}] =& \mathsf{Pr}[\mathcal{A} \text{ wins} \mid \mathsf{InvBal}] \cdot \mathsf{Pr}[\mathsf{InvBal}] + \mathsf{Pr}[\mathcal{A} \text{ wins} \mid \mathsf{TamPi}] \cdot \mathsf{Pr}[\mathsf{TamPi}] \\ &+ \mathsf{Pr}[\mathcal{A} \text{ wins} \mid \neg\mathsf{InvBal} \wedge \neg\mathsf{TamPi}] \cdot \mathsf{Pr}[\neg\mathsf{InvBal} \wedge \neg\mathsf{TamPi}]\end{aligned}$$

Clearly, if none of the above two events occurs, then $\mathcal{A}$ cannot have won, hence, $\mathsf{Pr}[\mathcal{A} \text{ wins} \mid \neg\mathsf{InvBal} \wedge \neg\mathsf{TamPi}] = 0$ and we ignore this term in the sequel. In contrast, if any of the two events occurs, $\mathcal{A}$ definitely wins. Hence, we can compress $\mathcal{A}$'s success probability to

$$\mathsf{Pr}[\mathcal{A} \text{ wins}] = \mathsf{Pr}[\mathsf{InvBal}] + \mathsf{Pr}[\mathsf{TamPi}].$$

Claim: $\mathsf{Pr}[\mathsf{InvBal}] \leq negl(\lambda)$*:* We start with the invalid ballot case, assuming that the proof π is accepted. Since π is accepted, all ballots accepted are deemed valid. This can however not be the case since $V \neq V''$. So at least one ballot B must be invalid even though it appears valid. Since there are only polynomially many potentially invalid ballots, we scan the above list, arguing why in the just described case, at least one of the underlying cryptographic primitives has been broken by $\mathcal{A}$. The success probability of the adversary $\mathcal{B}$ that can be constructed in those cases is the same as the success probability of $\mathcal{A}$, just lowered by the probability of guessing the correct potentially invalid ballot B.

- **P is valid with respect to B even though it should not.** This can have two reasons. Either P itself is a proof of a wrong statement, i.e., the public components are somehow wrong, or P is correct but the public components are incorrect. In the first case, $\mathcal{B}$ can use P to break the soundness of the zero-knowledge proof scheme with non-negligible probability, which contradicts the assumption that ZKP is sound. In the latter case, we have to look closer at the statement: P proves the statement

$$\begin{aligned}&\exists\{\tilde{\sigma}^j\}_{j\in I\cup\{0\}}, id, r, tkn, v, \sigma_{id}.\\ &\quad \top = \mathsf{vfy}(\sigma_{id}, vk_{\mathsf{PKI}}, id) \wedge\\ &\quad \top = \mathsf{vfy}(\tilde{\sigma}^0, vk_{\mathsf{CI}}, (id, r)) \wedge\\ &\quad \forall j \in I.\ \hat{\sigma}^j = \mathsf{vfy}_{\mathsf{start}}(\tilde{\sigma}^j, vk_{\mathsf{CI}}, (id, r)) \wedge\\ &\quad \hat{p} \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(id, svc)) \ \wedge\ \hat{tkn} = \mathsf{E}(ek_{\mathsf{SP}}, tkn)\\ &\quad \wedge\ \hat{v} \leftarrow \mathsf{E}(ek_{\mathsf{SP}}, v) \ \wedge\ v \in C\end{aligned}$$

  and is correct with respect to the public components. Nonetheless, the ballot is invalid. Hence, there must be an incorrect public component which is related to the PKI, the only honest party in the verifiability game instantiation for CR^{2}P. We can then use that component to break some property.

  - σ_{id} is a signature on an identity that has never been created by the PKI. In that case, we can use the knowledge extractor to extract σ_{id} along with id to break the unforgeability of the digital signature scheme. Also, in that case the entire ballot would be invalid since the identity behind it does not exist. This case can clearly not occur since it would mean a contradiction to the unforgeability of the signature scheme.
  - $\hat{p}$ is the encryption of another user's pseudonym, not the one of user U_j for which the ballot is deemed valid. In that case, we can construct an adversary $\mathcal{B}$ which breaks the uniqueness of service-specific pseudonyms. Since P is valid, $\hat{p}$ must be the encryption of $\mathsf{SSP}_{\mathbb{G}}(sk, svc)$ where $sk = sk_j$. Likewise, since $\hat{p}$ is the encryption of another user's pseudonym, we have that $\mathsf{SSP}_{\mathbb{G}}(sk_j, svc) = \mathsf{SSP}_{\mathbb{G}}(sk_k, svc)$ for some $k \neq j$, which is a contradiction the uniqueness of SSPs. Hence, also this case cannot occur.

  To conclude, none of the public components can be tampered with by $\mathcal{A}$ and P is valid. This part of the validity condition can, thus, not be violated with more than a negligible probability.

- **The partially verified signatures $\hat{\sigma}^j$ can be verified to the end by using the plaintexts of $\hat{oi}^j$ and $\hat{at}^j$, respectively, even though they should not.** This property is verified in the tallying process and the necessary proof of decryption for $\hat{oi}^j$ and $\hat{at}^j$ is contained in π. Since we assume π to be correct, it must be the case that the $\hat{\sigma}^j$ are also formed correct, otherwise the proof π would not succeed or

P would be wrong. Both are not the case, hence, this part of the validity condition cannot be violated by $\mathcal{A}$ without being detected.

- **v is encrypted in $\hat{v}$ even though it should not.** Notice that P shows the knowledge of some v' and that this v' is encrypted in $\hat{v}$, and additionally that it belongs to a set of valid rating choices C. Since P is correct, in order to validate this point, it can only be the case that $v' \neq v$. The decryption of the $\hat{v}$'s is done later in the tallying phase, accompanied by a proof of decryption, which is contained in π. We assume that π is correct. Evidently, it is true that $\hat{v}$ is decrypted to v in the tallying phase, otherwise v would not be in the final tally. Hence, due to the correctness of π and P, it must be the case that $v = v'$ and hence, $\mathcal{A}$ cannot invalidate the ballot by means of this validity criterion.

- **$\hat{tkn}$ is the encryption of U_j's token even though it should not.** Also this property is verified later in the tallying process and the correctness proof of that verification is contained in π. Since π is correct in the present case, it must be the case that $\mathcal{A}$ encrypted a token for user U_k which collides with the one of user U_j. Let us analyze how tkn is formed. We have

$$tkn = \mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(sk_j, svc)^{\sum_{\ell=1}^{m} b_\ell})$$

and at the same time

$$tkn = \mathsf{E}(ek_{\mathsf{SP}}, \mathsf{SSP}_{\mathbb{G}}(sk_k, svc)^{\sum_{\ell=1}^{m} b_\ell}),$$

which again forms a contradiction to the uniqueness of SSPs and can hence, only occur with negligible probability.

To conclude, we have shown that an invalid ballot can only be deemed valid with negligible probability and hence, $\Pr[\mathsf{InvBal}] \leq negl(\lambda)$.

Claim: $\Pr[\mathsf{TamPi}] \leq negl(\lambda)$: In this second case, we assume that π is erroneously accepted by the verifiability challenger, $V \neq V''$, but there is no invalid ballot, the rating of which ends up in V even though it should not. In order to show that this setup cannot happen with more than a negligible probability, we have to prove that π is partially formed of zero-knowledge proofs that can be used to break the soundness of the underlying proof scheme, thereby deriving a contradiction. We do that by scanning through the tallying procedure step by step, analyzing in how far $\mathcal{A}$ can manipulate the final tally and from that information extracting a proof that can be used to break the soundness.

Ballot verification: even though we stated above that no vote of an invalid ballot ends up in V, we have still to argue that the public components $\hat{oi}^j$ and $\hat{at}^j$ cannot have been tampered with unless $\mathcal{A}$ is able to create proofs for wrong statements. Suppose that there exists a ballot B, the vote of which ends up in V, and which is invalid due to incorrect public components $\hat{oi}^j$ and $\hat{at}^j$, i.e., if oi^j and at^j are the plaintexts contained in those components, then we have $\perp = \mathsf{vfy}_{\mathsf{fin}}(\hat{\sigma}^j, vk_{\mathsf{CI}}, at^j, oi^j)$ and still the ballot proceeds in the tallying process. Since the proof P in B is correct by assumption, it must be the case that $\mathcal{A}$ manipulates the decryption of $\hat{at}^j$ or $\hat{oi}^j$ such that $\top = \mathsf{vfy}_{\mathsf{fin}}(\hat{\sigma}^j, vk_{\mathsf{CI}}, at^j, oi^j)$. For instance, assume that $\hat{at}^j$ encrypted an attribute at^j which would not allow the respective user to access the service. However, the partially verified signature $\hat{\sigma}^j$ verifies on a different attribute at'^j that would allow the user to participate. Usually, the finishing verification would fail unless $\mathcal{A}$ manages to manipulate the proof of correct decryption. The same could also be done based on the opening information in an analogous way.

Otherwise the public components would be correct and $\mathcal{A}$ had no advantage in breaking the game. Since each decryptor generated by $\mathcal{A}$ for the distributed decryption is accompanied by a proof of correctness, it must be the case that one of those proofs is a proof of a wrong statement and $\mathcal{B}$ can use it to break the soundness of the underlying proof system. Since $\mathcal{A}$ wins with non-negligible probability, so does $\mathcal{B}$ (only lowered by a factor inverse proportional to the number of proofs created by $\mathcal{A}$), which is still non-negligible. That is a contradiction to the soundness of the zero-knowledge proof system. Hence, the public components $\hat{oi}^j$ and $\hat{at}^j$ are correct except with negligible probability.

Removing duplicates: duplicates are removed based on the tokens tkn, which are decrypted in the previous step. With the same argument as above (on the decryptors and their correctness proofs on the ciphertext rather than in the PET), we can show that all tokens $tkn_1, \ldots, tkn_n$ on BB (after decryption) must be the plaintexts encrypted in $\hat{tkn}_1, \ldots, \hat{tkn}_n$, otherwise a decryptor validity proof can be used to break the soundness of the proof system. In the duplicate elimination step, the service providers jointly decrypt the tokens, sort them, and remove ballots as dictated by RV. The decryption step happens equivalently to the previous step, hence, the same argument applies: $\mathcal{B}$ can use one decryptor validity proof to break the soundness of the proof system.

As a side remark, we could imagine an attack in which $\mathcal{A}$ creates an invalid ballot by copying a valid token from another valid ballot. If that were possible then that valid ballot would be excluded in the duplicate elimination step, leaving the invalid ballot with the same token. This is however prevented by the protocol due to the proof of correct encryption of the encrypted token $\hat{tkn}$, which is included in P and cannot be successfully recreated without knowing the underlying plaintext (the token tkn).

Mixing: $\mathcal{A}$ can potentially change V in this step by including (resp. removing) not yet existing (resp. existing) ballots in (resp. from) BB. This is possible when it is $\mathcal{A}$'s turn to mix the bulletin board, i.e., to apply a random permutation and a new randomness to every ciphertext. Since $\mathcal{A}$ wins the game with non-negligible advantage, it must be the case that it either exchanges a ballot on the board for something else and hence that ballot disappears, or that it duplicates a ballot on the board (for which another ballot disappears). In any case, if the disappeared ballot was valid or the duplicated or added ballot was valid, then V changes. Hence, $\mathcal{B}$ guesses a mixing proof of a corrupted service provider and submits this proof to the soundness challenger. Since $\mathcal{A}$ wins with non-negligible probability and $\mathcal{B}$ only forwards outputs of $\mathcal{A}$, so wins $\mathcal{B}$ with non-negligible probability, only lowered by the probability of guessing the correct service provider. This is a contradiction to the assumption that the mixing proof is sound.

Removing votes with invalid tokens: In this step, $\mathcal{A}$ can potentially invalidate (resp. validate) at least one valid (resp. invalid) ballot by modifying the outcome of the PET or applying an incorrect blinding factor. Since $\mathcal{A}$ wins with non-negligible probability, one of the two described cases must occur. Since each such computation is accompanied by a correctness proof, $\mathcal{B}$ can use one of the correctness proofs to break the soundness of the underlying proof system. The success probability carries over from $\mathcal{A}$ to $\mathcal{B}$ except that $\mathcal{B}$ also has to guess the right proof, which lowers the probability by a polynomial factor. This is still a contradiction to the assumption that the proof system is sound.

Notice that this step also captures attacks based on originally maliciously generated tokens: since all parties can be controlled by $\mathcal{A}$, no one knows whether the created tokens are valid or not. Note also that even though all users are registered successfully, no-one prevents $\mathcal{A}$ from re-registering the users' tokens after coercion since it controls all service providers. Wrong tokens, however, are detected in the present step since $\mathcal{A}$ has to relate its blinding of encrypted pseudonyms (which are proven to be correctly incorporated in a ballot B in the accompanying proof P, hence, they cannot have been tampered with; see also the previous claim) to the values B_i published initially on the bulletin board. Hence, at least the recomputed token is correctly computed, otherwise the correctness proof could be used to break soundness again. To conclude this point, no invalid token can survive this step without being detected.

Decrypting the votes: $\mathcal{A}$ can only change V from the correct one V'' by decrypting the ratings $\hat{v}_i$ incorrectly. Since $\mathcal{A}$ wins the game with non-negligible probability, it must be the case that there exists an encrypted rating $\hat{v}$ for which the plaintext has been changed during the decryption procedure. Since every decryptor is accompanied by a correctness proof, $\mathcal{B}$ can use that proof to break the soundness of the underlying proof system with also non-negligible probability, which is a contradiction to the assumption.

To conclude this point, we have shown that in any step of the Tally protocol, if $\mathcal{A}$ wants to manipulate V such that $V \neq V''$, then we can construct an adversary $\mathcal{B}$ against the soundness of the zero-knowledge proof system. That is a contradiction and concludes the proof for this step. Hence, $\Pr[\mathsf{TamPi}] \leq negl(\lambda)$.

To conclude overall, we have shown that

$$\Pr[\mathsf{InvBal}] \leq negl(\lambda)$$

and

$$\Pr[\mathsf{TamPi}] \leq negl(\lambda).$$

Consequently,

$$\mathsf{Pr}\left[\mathcal{A} \text{ wins}\right] \leq negl(\lambda) + negl(\lambda) = negl(\lambda)$$

and that concludes our proof. □

Bibliography

[1] Ben Adida. Helios: Web-based Open-audit Voting. In *Proc. USENIX Security Symposium (USENIX'08)*, pages 335–348. USENIX Association, 2008. (Cited on pages 93, 103, and 196.)

[2] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private Information Retrieval for Everyone. In *Proc. Privacy Enhancing Technologies Symposium (PETS'16)*, pages 155–174. De Gruyter, 2016. (Cited on pages 50 and 131.)

[3] Miklós Ajtai. Oblivious RAMs Without Cryptographic Assumptions. In *Proc. ACM Symposium on Theory of Computing (STOC'10)*, pages 181–190. ACM, 2010. (Cited on pages 4 and 57.)

[4] Istemi Ekin Akkus, Ruichuan Chen, Michaela Hardt, Paul Francis, and Johannes Gehrke. Non-tracking Web Analytics. In *Proc. Conference on Computer and Communications Security (CCS'12)*, pages 687–698. ACM, 2012. (Cited on page 14.)

[5] Amazon Help. About Amazon Verified Purchase Reviews. Online at `https://www.amazon.com/gp/help/customer/display.html/ref=hp_20079100_verifiedreviews?nodeId=201145140`. (Cited on page 6.)

[6] Elli Androulaki, Seung Geol Choi, Steven M. Bellovin, and Tal Malkin. Reputation Systems for Anonymous Networks. In *Proc. Privacy Enhancing Technologies Symposium (PETS'08)*, pages 202–218. Springer Verlag, 2008. (Cited on pages 111 and 112.)

[7] I Get Paid To Write Fake Reviews For Amazon, 2017. Online at `www.cracked.com`. (Cited on page 84.)

[8] Daniel Apon, Jonathan Katz, Elaine Shi, and Aishwarya Thiruvengadam. Verifiable Oblivious Storage. In *Proc. Practice and Theory in Public Key Cryptography (PKC'14)*, LNCS, pages 131–148. Springer Verlag, 2014. (Cited on pages 4 and 57.)

[9] Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, and Ramarathnam Venkatesan. Orthogonal Security With Cipherbase. In *Proc. Biennial Conference on Innovative Data Systems Research (CIDR'13)*, 2013. (Cited on page 64.)

[10] Roberto Araújo, Amira Barki, Solenn Brunet, and Jacques Traoré. Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistant. In *Proc. Conference on*

Financial Cryptography and Data Security (FC'16), LNCS, pages 224–232. Springer Verlag, 2016. (Cited on pages 112 and 113.)

[11] Roberto Araújo, Narjes Ben Rajeb, Riadh Robbana, Jacques Traoré, and Souheib Youssfi. Towards Practical and Secure Coercion-Resistant Electronic Elections. In *Proc. International Conference on Cryptology and Network Security (CANS'10)*, pages 278–297. Springer Verlag, 2010. (Cited on pages 112 and 113.)

[12] Roberto Araújo and Jacques Traoré. A Practical Coercion Resistant Voting Scheme Revisited. In *Proc. Conference on E-Voting and Identity (VoteID'13)*, LNCS, pages 193–209. Springer Verlag, 2013. (Cited on pages 112 and 113.)

[13] Giuseppe Ateniese and Breno de Medeiros. On the Key Exposure Problem in Chameleon Hashes. In *Proc. International Conference on Security in Communication Networks (SCN'04)*, LNCS, pages 165–179. Springer Verlag, 2004. (Cited on pages 50, 126, and 132.)

[14] Man H. Au, Apu Kapadia, and Willy Susilo. BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation. In *Proc. Annual Network & Distributed System Security Symposium (NDSS'12)*. Internet Society, 2012. (Cited on pages 84 and 112.)

[15] Julian Backes, Stefan Lorenz, and Kim Pecina. Zero-knowledge Library. online at `github.com/peloba/zk-library`. (Cited on page 50.)

[16] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus. In *Proc. IEEE Symposium on Computer Security Foundations (CSF'08)*, pages 195–209. IEEE Press, 2008. (Cited on pages 90 and 112.)

[17] Michael Backes, Stefan Lorenz, Matteo Maffei, and Kim Pecina. Anonymous Webs of Trust. In *Proc. Privacy Enhancing Technologies Symposium (PETS'10)*, LNCS, pages 130–148. Springer Verlag, 2010. (Cited on page 34.)

[18] Michael Backes, Matteo Maffei, and Kim Pecina. Automated Synthesis of Privacy-Preserving Distributed Applications. In *Proc. Annual Network & Distributed System Security Symposium (NDSS'12)*. Internet Society, 2012. (Cited on pages 34, 84, and 112.)

[19] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous Credentials Light. In *Proc. Conference on Computer and Communications Security (CCS'13)*, pages 1087–1098. ACM, 2013. (Cited on page 34.)

[20] Michael Barbaro and Steve Edermarch. At Trump University, Students Recall Pressure to Give Positive Reviews, 2016. Online at `www.nytimes.com`. (Cited on page 84.)

[21] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical Multi-candidate Election System. In *Proc. ACM Symposium on Principles of Distributed Computing (PODC'01)*, pages 274–283. ACM, 2001. (Cited on page 112.)

[22] Stephanie Bayer and Jens Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'12)*, LNCS, pages 263–280. Springer Verlag, 2012. (Cited on pages 12, 37, 41, 50, 99, 108, 110, 132, and 185.)

[23] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In *Proc. Advances in Cryptology (CRYPTO'09)*, volume 5677 of *LNCS*, pages 108–125. Springer Verlag, 2009. (Cited on pages 84 and 112.)

[24] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and Noninteractive Anonymous Credentials. In *Proc. Conference on Theory of Cryptography (TCC'08)*, pages 356–374. Springer Verlag, 2008. (Cited on pages 84 and 112.)

[25] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Proc. Advances in Cryptology (CRYPTO'98)*, LNCS, pages 26–45. Springer Verlag, 1998. (Cited on page 23.)

[26] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. DupLESS: Server-Aided Encryption for Deduplicated Storage. In *Proc. USENIX Security Symposium (USENIX'13)*, pages 179–194. USENIX Association, 2013. (Cited on page 64.)

[27] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable Delegation of Computation Over Large Datasets. In *Proc. Advances in Cryptology (CRYPTO'11)*, LNCS, pages 111–131. Springer Verlag, 2011. (Cited on page 60.)

[28] Josh Benaloh. Simple Verifiable Elections. In *Proc. USENIX/Accurate Electronic Voting Technology Workshop (EVT'06)*, pages 5–5. USENIX Association, 2006. (Cited on pages 93 and 105.)

[29] Vincent Bindschaedler, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, and Yan Huang. Practicing Oblivious Access on Cloud Storage: The Gap, the Fallacy, and the New Way Forward. In *Proc. Conference on Computer and Communications Security (CCS'15)*, pages 837–849. ACM, 2015. (Cited on pages 4, 11, 24, 32, 57, and 59.)

[30] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on Randomizable Ciphertexts. In *Proc. Practice and Theory in Public Key Cryptography (PKC'11)*, LNCS, pages 403–422. Springer Verlag, 2011. (Cited on page 96.)

[31] Johannes Blömer, Jakob Juhnke, and Christina Kolb. Anonymous and Publicly Linkable Reputation Systems. In *Proc. Financial Cryptography and Data Security (FC'15)*, pages 478–488. Springer Verlag, 2015. (Cited on pages 111 and 112.)

[32] BlueKrypt. Cryptograhpic Key Length Recommendation. online at `www.keylength.com`. (Cited on pages 50, 51, and 110.)

[33] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo Voting: Secure and Coercion-free Voting Using a Trusted Random Number Generator. In *Proc. Conference on E-voting and Identity (VoteID'07)*, pages 111–124. Springer Verlag, 2007. (Cited on page 112.)

[34] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-Preserving Symmetric Encryption. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'09)*, LNCS, pages 224–241. Springer Verlag, 2009. (Cited on pages 64, 65, and 68.)

[35] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In *Proc. Advances in Cryptology (CRYPTO'11)*, pages 578–595. Springer Verlag, 2011. (Cited on page 64.)

[36] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'15)*, pages 563–594. Springer Verlag, 2015. (Cited on page 64.)

[37] Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious Parallel RAM and Applications. In *Proc. Theory of Cryptography (TCC'16)*, LNCS. Springer Verlag, 2016. (Cited on pages 58 and 59.)

[38] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *Proc. Practice and Theory in Public Key Cryptography (PKC'09)*, LNCS, pages 481–500. Springer Verlag, 2009. (Cited on page 47.)

[39] Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Proc. Advances in Cryptology (CRYPTO'02)*, volume 2442 of *LNCS*, pages 61–76. Springer Verlag, 2002. (Cited on pages 84 and 112.)

[40] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report 260, ETH Zürich, Institute for Theoretical Computer Science, March 1997. (Cited on pages 108 and 183.)

[41] Bogdan Carbunar and Radu Sion. Regulatory Compliant Oblivious RAM. In *Proc. International Conference on Applied Cryptography and Network Security (ACNS'10)*, LNCS, pages 456–474. Springer Verlag, 2010. (Cited on pages 4 and 57.)

[42] Angelo De Caro. jPBC - Java Library for Pairing Based Cryptography. online at `http://gas.dia.unisa.it/projects/jpbc/`. (Cited on page 50.)

[43] Inmaculada Carrión Señor, Luis José Fernández-Alemán, and Ambrosio Toval. Are Personal Health Records Safe? A Review of Free Web-Accessible Personal Health Record Privacy Policies. *Journal of Medical Internet Research*, 14(4), 2012. (Cited on page 60.)

[44] J. Lawrence Carter and Mark N. Wegman. Universal Classes of Hash Functions (Extended Abstract). In *Proc. ACM Symposium on Theory of Computing (STOC'77)*, pages 106–112. ACM, 1977. (Cited on pages 12, 43, and 142.)

[45] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *Proc. Conference on Computer and Communications Security (CCS'16)*, pages 1614–1625. ACM, 2016. (Cited on page 96.)

[46] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end Verifiability for Optical Scan Election Systems Using Invisible Ink Confirmation Codes. In *Proc. Conference on Electronic Voting Technology (EVT'08)*, pages 14:1–14:13. USENIX Association, 2008. (Cited on page 112.)

[47] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty Unconditionally Secure Protocols. In *Proc. ACM Symposium on Theory of Computing (STOC'88)*, pages 11–19. ACM, 1988. (Cited on page 31.)

[48] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A Practical Voter-verifiable Election Scheme. In *Proc. European Symposium on Research in Computer Security (ESORICS'05)*, LNCS, pages 118–139. Springer Verlag, 2005. (Cited on page 112.)

[49] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981. (Cited on pages 37 and 128.)

[50] Binyi Chen, Huijia Lin, and Stefano Tessaro. Oblivious Parallel RAM: Improved Efficiency and Generic Constructions. In *Proc. Theory of Cryptography (TCC'16)*, LNCS. Springer Verlag, 2016. (Cited on pages 58 and 59.)

[51] Ruichuan Chen, Istemi Ekin Akkus, and Paul Francis. SplitX: High-Performance Private Analytics. In *Proc. of the ACM SIGCOMM 2013*, pages 315–326. ACM, 2013. (Cited on page 14.)

[52] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. Practical Order-Revealing Encryption with Limited Leakage. In Thomas Peyrin, editor, *Proc. Fast Software Encryption (FSE'16)*, pages 474–493. Springer Verlag, 2016. (Cited on pages 64 and 65.)

[53] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private Information Retrieval. *Journal of the ACM*, 45(6):965–981, November 1998. (Cited on pages 4, 11, 22, and 128.)

[54] Jeremy Clark and Urs Hengartner. Selections: Internet Voting with Over-the-shoulder Coercion-resistance. In *Proc. Financial Cryptography and Data Security (FC'11)*, LNCS, pages 47–61. Springer Verlag, 2012. (Cited on page 112.)

[55] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *Proc. IEEE Symposium on Security & Privacy (S&P'08)*, pages 354–368. IEEE Press, 2008. (Cited on pages 85, 86, 106, 108, 112, 184, and 185.)

[56] Véronique Cortier, Fabienne Eigner, Steve Kremer, Matteo Maffei, and Cyrille Wiedling. Type-Based Verification of Electronic Voting Protocols. In *Proc. Principles of Security and Trust (POST'15)*, volume 9036 of *LNCS*, pages 303–323. Springer Verlag, 2015. (Cited on pages 93 and 105.)

[57] Roberto Di Cosmo. On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack, 2007. Online at `http://hal.archives-ouvertes.fr/hal-00142440`. (Cited on page 106.)

[58] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Proc. Advances in Cryptology (CRYPTO'94)*, LNCS, pages 174–187. Springer Verlag, 1994. (Cited on pages 50, 108, 131, 183, and 185.)

[59] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-authority Election Scheme. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'97)*, LNCS, pages 103–118. Springer Verlag, 1997. (Cited on pages 108, 132, and 183.)

[60] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Proc. Advances in Cryptology (CRYPTO'98)*, LNCS, pages 13–25. Springer Verlag, 1998. (Cited on pages 50, 122, and 131.)

[61] Chris Crum. Just How Bad Is Yelp's Fake Review Problem? `www.webpronews.com`, 2014. (Cited on page 84.)

[62] Chris Culnane and Steve Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In *Proc. Symposium on Computer Security Foundations (CSF'14)*, pages 169–183. IEEE Press, 2014. (Cited on pages 27, 45, and 86.)

[63] Joan Daemen and Vincent Rijmen. *The Design of Rijndael, AES - The Advanced Encryption Standard*. Springer Verlag, 2002. (Cited on pages 50 and 131.)

[64] David Daglish and Norm Archer. Electronic Personal Health Record Systems: A Brief Review of Privacy, Security, and Architectural Issues. *World Congress on Privacy, Security, Trust and the Management of e-Business*, pages 110–120, 2009. (Cited on page 60.)

[65] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly Secure Oblivious RAM Without Random Oracles. In *Proc. Theory of Cryptography (TCC'11)*, LNCS, pages 144–163. Springer Verlag, 2011. (Cited on pages 4 and 57.)

[66] Jonathan Dautrich, Emil Stefanov, and Elaine Shi. Burst ORAM: Minimizing ORAM Response Times for Bursty Access Patterns. In *Proc. USENIX Security Symposium (USENIX'14)*, pages 749–764. USENIX Association, 2014. (Cited on pages 4 and 57.)

[67] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Private Data Indexes for Selective Access to Outsourced Data. In *Proc. Annual ACM Workshop on Privacy in the Electronic Society (WPES'11)*, pages 69–80. ACM, 2011. (Cited on page 60.)

[68] Stephanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *Proc. IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 28–42. IEEE Press, 2006. (Cited on page 112.)

[69] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proc. Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12. ACM, 1987. (Cited on pages 23, 24, and 160.)

[70] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976. (Cited on pages 99 and 122.)

[71] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proc. USENIX Security Symposium (USENIX'04)*, pages 303–320. USENIX Association, 2004. (Cited on page 34.)

[72] Changyu Dong and Liqun Chen. A Fast Single Server Private Information Retrieval Protocol with Low Communication Cost. In *Proc. European Symposium on Research in Computer Security (ESORICS'14)*, volume 8712 of *LNCS*, pages 380–399. Springer Verlag, 2014. (Cited on page 31.)

[73] F. Betül Durak, Thomas M. DuBuisson, and David Cash. What Else is Revealed by Order-Revealing Encryption? In *Proc. Conference on Computer and Communications Security (CCS'16)*, pages 1155–1166. ACM, 2016. (Cited on page 64.)

[74] Karen Easterbrook, Kevin Kane, Lan Nguyen, Christian Paquin, and Greg Zaverucha. U-Prove, 2012. `http://research.microsoft.com/en-us/projects/u-prove/`. (Cited on pages 84 and 112.)

[75] Elektronische Gesundheitskarte, 2017. Online at `https://www.bundesgesundheitsministerium.de/themen/krankenversicherung/egk.html`. (Cited on page 1.)

[76] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Proc. Advances in Cryptology (CRYPTO'84)*, LNCS, pages 10–18. Springer Verlag, 1985. (Cited on pages 41, 50, 99, 108, 131, and 181.)

[77] Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, and Roberto Tamassia. Dynamic Provable Data Possession. In *Proc. Conference on Computer and Communications Security (CCS'09)*, pages 213–222. ACM, 2009. (Cited on page 60.)

[78] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proc. Advances in Cryptology (CRYPTO'86)*, pages 186–194. Springer Verlag, 1987. (Cited on pages 42, 43, 108, and 185.)

[79] Lisa Fichenscher. Scammers elude Amazon crackdown on fake reviews with new tricks, 2017. Online at `http://nypost.com`. (Cited on page 84.)

[80] Décio Luiz Gazzoni Filho and Paulo Sérgio Licciardi Messeder Barreto. Demonstrating Data Possession and Uncheatable Data Transfer. Cryptology ePrint Archive, Report 2006/150, 2006. `http://eprint.iacr.org/`. (Cited on page 60.)

[81] Martin Franz, Peter Williams, Bogdan Carbunar, Stefan Katzenbeisser, Andreas Peter, Radu Sion, and Miroslava Sotakova. Oblivious Outsourced Storage with Delegation. In *Proc. Financial Cryptography and Data Security (FC'11)*, pages 127–140. Springer Verlag, 2011. (Cited on pages 22, 23, 58, and 59.)

[82] David Mandell Freeman. Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'10)*, LNCS, pages 44–61. Springer Verlag, 2010. (Cited on pages 131 and 132.)

[83] Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin. Coercion Resistant End-to-end Voting. In *Proc. Conference on Financial Cryptography and Data Security (FC'09)*, pages 344–361. Springer Verlag, 2009. (Cited on page 112.)

[84] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. In *Proc. Annual Network & Distributed System Security Symposium (NDSS'14)*. Internet Society, 2014. (Cited on pages 84 and 112.)

[85] Craig Gentry and Brent Waters. Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts). In *Proc. Conference on the Theory and Applications of*

Cryptographic Techniques (EUROCRYPT'09), LNCS, pages 171–188. Springer Verlag, 2009. (Cited on pages 47, 50, 125, and 131.)

[86] Gesundheitstelematikgesetz, 2017. Online at `www.ris.bka.gv.at`. (Cited on page 1.)

[87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play ANY Mental Game. In *Proc. ACM Symposium on Theory of Computing (STOC'87)*, pages 218–229. ACM, 1987. (Cited on page 31.)

[88] Oded Goldreich and Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM*, 43(3):431–473, May 1996. (Cited on pages 4, 11, and 57.)

[89] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information. In *Proc. ACM Symposium on Theory of Computing (STOC'82)*, pages 365–377. ACM, 1982. (Cited on pages 23, 121, and 122.)

[90] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 127.)

[91] Michael T. Goodrich and Michael Mitzenmacher. Privacy-Preserving Access of Outsourced Data via Oblivious RAM Simulation. In *Proc. International Conference on Automata, Languages and Programming (ICALP'11)*, LNCS, pages 576–587. Springer Verlag, 2011. (Cited on pages 4 and 57.)

[92] Michael T. Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Privacy-Preserving Group Data Access via Stateless Oblivious RAM Simulation. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*, pages 157–167. Society for Industrial and Applied Mathematics, 2012. (Cited on page 58.)

[93] Jens Groth. Non-interactive Zero-knowledge Arguments for Voting. In *Proc. International Conference on Applied Cryptography and Network Security (ACNS'05)*, LNCS, pages 467–482. Springer Verlag, 2005. (Cited on pages 108 and 183.)

[94] Jens Groth and Amit Sahai. Efficient Noninteractive Proof Systems for Bilinear Groups. *SIAM Journal on Computing*, 41(5):1193–1232, 2012. (Cited on pages 12, 50, 131, 132, 133, and 136.)

[95] Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking Web Applications Built On Top of Encrypted Data. In *Proc. Conference on Computer and Communications Security (CCS'16)*, pages 1353–1364. ACM, 2016. (Cited on page 64.)

[96] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-Abuse Attacks against Order-Revealing Encryption. In

Proc. IEEE Symposium on Security & Privacy (S&P'17). IEEE Press, 2017. (Cited on pages 5, 64, 65, and 66.)

[97] Omar Hasan, Lionel Brunie, and Elisa Bertino. Preserving Privacy of Feedback Providers in Decentralized Reputation Systems. *Computer Securурity*, 31(7):816–826, October 2012. (Cited on pages 111 and 112.)

[98] Warren He, Devdatta Akhawe, Sumeet Jain, Elaine Shi, and Dawn Song. ShadowCrypt: Encrypted Web Applications for Everyone. In *Proc. Conference on Computer and Communications Security (CCS'14)*, pages 1028–1039. ACM, 2014. (Cited on page 64.)

[99] James Heather and David Lundin. The Append-Only Web Bulletin Board. In *Proc. USENIX Conference on File and Storage Technologies (FAST'09)*, pages 242–256. Springer Verlag, 2009. (Cited on pages 27, 45, and 86.)

[100] James Heather and Steve Schneider. A Formal Framework for Modelling Coercion Resistance and Receipt Freeness. In *Proc. International Symposium on Formal Methods (FM'12)*, 2012. (Cited on page 112.)

[101] Health insurance portability and accountability act, 2017. Online at `https://www.hhs.gov/hipaa/index.html`. (Cited on page 1.)

[102] Martin Hirt and Kazue Sako. Efficient Receipt-free Voting Based on Homomorphic Encryption. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'00)*, LNCS, pages 539–556. Springer Verlag, 2000. (Cited on page 112.)

[103] Yizhou Huang and Ian Goldberg. Outsourced Private Information Retrieval with Pricing and Access Control. In *Proc. Annual ACM Workshop on Privacy in the Electronic Society (WPES'13)*. ACM, 2013. (Cited on pages 57 and 59.)

[104] Thomas Icart. How to Hash into Elliptic Curves. In *Proc. Advances in Cryptology (CRYPTO'09)*, pages 303–316, 2009. (Cited on page 182.)

[105] Identity Mixer, 2005. `http://idemix.wordpress.com/`. (Cited on pages 84 and 112.)

[106] Alexander Iliev and Sean W. Smith. Protecting Client Privacy with Trusted Computing at the Server. *IEEE Security and Privacy*, 3(2):20–28, March 2005. (Cited on pages 4 and 57.)

[107] Mohammad Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access Pattern Disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *Proc. Annual Network & Distributed System Security Symposium (NDSS'12)*. Internet Society, 2012. (Cited on pages 2 and 10.)

[108] Markus Jakobsson and Ari Juels. Millimix: Mixing in Small Batches. Technical Report 99-33, DIMACS, 1999. (Cited on pages 41, 50, 131, and 132.)

[109] Markus Jakobsson and Ari Juels. Mix and Match: Secure Function Evaluation via Ciphertexts. In *Proc. Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'00)*, LNCS, pages 162–177. Springer Verlag, 2000. (Cited on pages 99 and 108.)

[110] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In *Proc. USENIX Security Symposium (USENIX'02)*, pages 339–353. USENIX Association, 2002. (Cited on pages 41, 99, 108, and 185.)

[111] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'96)*, LNCS, pages 143–154, Berlin, Heidelberg, 1996. Springer Verlag. (Cited on pages 99, 108, and 183.)

[112] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *Proc. ACM Workshop on Privacy in the Electronic Society (WPES'05)*, pages 61–70. ACM, 2005. (Cited on pages 84, 86, 93, and 112.)

[113] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'08)*, pages 146–162. Springer Verlag, 2008. (Cited on pages 12, 47, 50, 123, 131, 132, and 133.)

[114] Florian Kerschbaum. A Verifiable, Centralized, Coercion-free Reputation System. In *Proc. ACM Workshop on Privacy in the Electronic Society (WPES'09)*, pages 61–70. ACM, 2009. (Cited on pages 111 and 112.)

[115] Florian Kerschbaum. Frequency-Hiding Order-Preserving Encryption. In *Proc. Conference on Computer and Communications Security (CCS'15)*, pages 656–667. ACM, 2015. (Cited on pages 5, 64, 65, 66, 67, 68, 69, 71, 72, 74, 75, 76, 79, and 116.)

[116] Florian Kerschbaum and Axel Schroepfer. Optimal Average-Complexity Ideal-Security Order-Preserving Encryption. In *Proc. Conference on Computer and Communications Security (CCS'14)*, pages 275–286. ACM, 2014. (Cited on pages 64 and 68.)

[117] Shahram Khazaei and Douglas Wikström. Randomized Partial Checking Revisited. In *Proc. International Conference on Topics in Cryptology (CT-RSA'13)*, pages 115–128. Springer Verlag, 2013. (Cited on page 99.)

[118] Beom Heyn Kim and David Lie. Caelus: Verifying the Consistency of Cloud Services with Battery-Powered Devices. In *Proc. IEEE Symposium on Security & Privacy (S&P'15)*, pages 880–896. IEEE Press, 2015. (Cited on pages 23, 24, and 59.)

[119] Andreas Kokoschka, Ronald Petrlic, and Christoph Sorge. A Reputation System supporting Unlinkable, yet Authorized Expert Ratings. In *Proc. International Conference on Selected Areas in Cryptography (SAC'15)*. ACM, 2015. (Cited on pages 111 and 112.)

[120] Merel Koning, Paulan Korenhof, Gergely Alpár, and Jaap-Henk Hoepman. The ABC of ABC - An Analysis of Attribute-Based Credentials in the Light of Data Protection, Privacy and Identity. In *Proc. Privacy Enhancing Technologies Symposium (PETS'14)*. De Gruyter, 2014. (Cited on page 84.)

[121] Priyanka Korde, Vijay Panwar, and Sneha Kalse. Securing Personal Health Records in Cloud using Attribute Based Encryption. *International Journal of Engineering and Advanced Technology*, 2013. (Cited on page 60.)

[122] Ralf Küsters and Tomasz Truderung. Security Analysis of Re-Encryption RPC Mix Nets. In *Proc. European Symposium on Security and Privacy (EuroS&P'16)*. IEEE Press, 2016. (Cited on page 99.)

[123] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and Relationship to Verifiability. In *Proc. Conference on Computer and Communications Security (CCS'10)*, pages 526–535. ACM, 2010. (Cited on page 20.)

[124] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Proving Coercion-resistance of Scantegrity II. In *Proc. International Conference on Information and Communications Security (ICICS'10)*, pages 281–295. Springer Verlag, 2010. (Cited on page 112.)

[125] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A Game-based Definition of Coercion Resistance and Its Applications. *Journal of Computer Security*, 20(6):709–764, November 2012. (Cited on page 112.)

[126] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *Proc. IEEE Symposium on Security & Privacy (S&P'12)*, pages 395–409. IEEE Press, 2012. (Cited on page 105.)

[127] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking. In *Proc. IEEE Symposium on Security & Privacy (S&P'14)*, pages 343–358. IEEE Press, 2014. (Cited on page 99.)

[128] Paul Lajoie-Mazenc, Emmanuelle Anceaume, Gilles Guette, Thomas Sirvent, and Valérie Viet Triem Tong. Efficient Distributed Privacy-Preserving Reputation Mechanism Handling Non-Monotonic Ratings, 2015. Online at `https://hal.archives-ouvertes.fr/hal-01104837v2`. (Cited on pages 111 and 112.)

[129] Billy Lau, Simon Chung, Chengyu Song, Yeongjin Jang, Wenke Lee, and Alexandra Boldyreva. Mimesis Aegis: A Mimicry Privacy Shield–A System's Approach to Data Privacy on Public Cloud. In *Proc. USENIX Security Symposium (USENIX'14)*, pages 33–48. USENIX Association, 2014. (Cited on page 64.)

[130] Margaret Young Levi. Retention of Paper Medical Records After Converting to Electronic Health Records, 2013. Online at `https://wyatthitechlaw.com`. (Cited on page 1.)

[131] Kevin Lewi and David J. Wu. Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds. In *Proc. Conference on Computer and Communications Security (CCS'16)*, pages 1167–1178. ACM, 2016. (Cited on page 64.)

[132] Ming Li, Shucheng Yu, Kui Ren, and Wenjing Lou. Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-owner Settings. In *SECURECOMM'10*, 2010. (Cited on page 60.)

[133] Yehuda Lindell and Benny Pinkas. An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'07)*, LNCS, pages 52–78. Springer Verlag, 2007. (Cited on page 31.)

[134] Yehuda Lindell and Benny Pinkas. A Proof of Security of Yao's Protocol for Two-Party Computation. *Journal of Cryptology*, 22(2):161–188, April 2009. (Cited on pages 122 and 131.)

[135] Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. Securing the e-Health Cloud. In *Proc. ACM International Health Informatics Symposium (IHI'10)*, pages 220–229. ACM, 2010. (Cited on page 60.)

[136] Jacob R. Lorch, Bryan Parno, James Mickens, Mariana Raykova, and Joshua Schiffman. Shroud: Ensuring Private Access to Large-scale Data in the Data Center. In *Proc. USENIX Conference on File and Storage Technologies (FAST'13)*, pages 199–214. USENIX Association, 2013. (Cited on pages 4, 11, 57, 58, and 59.)

[137] Ben Lynn. PBC - C Library for Pairing Based Cryptography. online at `http://crypto.stanford.edu/pbc/`. (Cited on page 50.)

[138] Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Krste Asanovic, John Kubiatowicz, and Dawn Song. PHANTOM: Practical Oblivious Computation in a Secure Processor. In *Proc. Conference on Computer and Communications Security (CCS'13)*, pages 311–324. ACM, 2013. (Cited on pages 4 and 57.)

[139] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Brief Announcement: Towards Security and Privacy for Outsourced Data in the Multi-Party Setting. In *Proc. Symposium on Principles of Distributed Computing (PODC'14)*. ACM, 2014. (Cited on page vii.)

[140] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. GORAM: Privacy, Access Control, and Verifiability in Group Outsourced Storage. Full version, online at `http://www.sps.cs.uni-saarland.de/publications/goram.pdf`, 2014. (Cited on page 131.)

[141] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Privacy and Access Control for Outsourced Personal Records. In *Proc. IEEE Symposium on Security & Privacy (S&P'15)*. IEEE Press, 2015. (Cited on page vii.)

[142] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Maliciously Secure Multi-Client ORAM. In *Proc. International Conference on Applied Cryptography and Network Security (ACNS'17)*, LNCS. Springer Verlag, 2017. (Cited on page vii.)

[143] Matteo Maffei, Kim Pecina, and Manuel Reinert. Security and Privacy by Declarative Design. In *Proc. Symposium on Computer Security Foundations (CSF'13)*, pages 81–96. IEEE Press, 2013. (Cited on pages 34, 84, 99, 112, 127, and 182.)

[144] Matteo Maffei, Kim Pecina, Manuel Reinert, and Ahmed Salem. CR^{2}P: Coercion-Resistant Rating Platforms. 2017. (Cited on page vii.)

[145] Matteo Maffei, Manuel Reinert, and Dominique Schröder. On the Security of Frequency-Hiding Order-Preserving Encryption. In *Proc. International Conference on Cryptology and Network Security (CANS'17)*, LNCS. Springer Verlag, 2017. (Cited on page vii.)

[146] Travis Mayberry, Erik-Oliver Blass, and Agnes Hui Chan. Efficient Private File Retrieval by Combining ORAM and PIR. In *Proc. Annual Network & Distributed System Security Symposium (NDSS'14)*. Internet Society, 2013. (Cited on page 57.)

[147] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 84(5):1234–1243, 2001. (Cited on pages 50, 108, and 180.)

[148] Vincent Naessens, Liesje Demuynck, and Bart De Decker. A Fair Anonymous Submission and Review System. In *Proc. Communications and Multimedia Security (CMS'06)*, volume 4237 of *LNCS*, pages 43–53. Springer Verlag, 2006. (Cited on pages 111 and 112.)

[149] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference Attacks on Property-Preserving Encrypted Databases. In *Proc. Conference on Computer and Communications Security (CCS'15)*, pages 644–655. ACM, 2015. (Cited on page 64.)

[150] Nielsen. Nielsen: Global consumers' trust in 'earned' advertising grows in importance, 2012. Online at `www.nielsen.com`. (Cited on pages 3 and 5.)

[151] Office of the privacy commissioner of Canada, 2017. Online at `www.priv.gc.ca/en`. (Cited on page 1.)

[152] Rafail Ostrovsky and William E. Skeith III. Algebraic Lower Bounds for Computing on Encrypted Data. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(022), 2007. (Cited on pages 22 and 23.)

[153] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proc. Conference on the Theory and Applications of Cryptographic*

Techniques (EUROCRYPT'99), volume 1592 of *LNCS*, pages 223–238. Springer Verlag, 1999. (Cited on pages 41, 99, 108, 182, and 183.)

[154] Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi. Streaming Authenticated Data Structures . In *Proc. Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'13)*, 2013. (Cited on page 60.)

[155] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Proc. Advances in Cryptology (CRYPTO'91)*, LNCS, pages 129–140. Springer Verlag, 1992. (Cited on page 99.)

[156] Sarah Perez. Amazon sues more sellers for buying fake reviews, 2017. Online at `https://techcrunch.com`. (Cited on page 84.)

[157] Ronald Petrlic, Sascha Lutters, and Christoph Sorge. Privacy-preserving Reputation Management. In *Proc. International Conference on Selected Areas in Cryptography (SAC'14)*, pages 1712–1718. ACM, 2014. (Cited on pages 111 and 112.)

[158] Franziska Pingel and Sandra Steinbrecher. Multilateral Secure Cross-Community Reputation Systems for Internet Communities. In *Proc. Trust and Privacy in Digital Business (TRUSTBUS'08)*, volume 5185 of *LNCS*, pages 69–78. Springer Verlag, 2008. (Cited on pages 111 and 112.)

[159] Benny Pinkas and Tzachy Reinman. Oblivious RAM Revisited. In *Proc. Advances in Cryptology (CRYPTO'10)*, LNCS, pages 502–519. Springer Verlag, 2010. (Cited on pages 4, 10, and 57.)

[160] David Pointcheval and Olivier Sanders. Short Randomizable Signatures. In *Proc. International Conference on Topics in Cryptology (CT-RSA'16)*, volume 9610 of *LNCS*, pages 111–126. Springer Verlag, 2016. (Cited on page 108.)

[161] Raluca Ada Popa, Frank H. Li, and Nickolai Zeldovich. An Ideal-Security Protocol for Order-Preserving Encoding. In *Proc. IEEE Symposium on Security & Privacy (S&P'13)*, pages 463–477. IEEE Press, 2013. (Cited on pages 64 and 68.)

[162] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Proc. ACM Symposium on Operating Systems Principles (SOSP'11)*, pages 85–100. ACM, 2011. (Cited on page 64.)

[163] Raluca Ada Popa, Emily Stark, Steven Valdez, Jonas Helfer, Nickolai Zeldovich, and Hari Balakrishnan. Building Web Applications on Top of Encrypted Data Using Mylar. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI'14)*, pages 157–172. USENIX Association, 2014. (Cited on page 64.)

[164] Rainey Reitman. Medical Justice: Stifling Speech of Patients with a Touch of "Privacy Blackmail". `www.eff.org`, 2011. (Cited on pages 3, 5, and 84.)

[165] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. (Cited on pages 50 and 132.)

[166] Reid J. Robinson. How big is the human genome? Online at `https://medium.com/precision-medicine/how-big-is-the-human-genome-e90caa3409b0`. (Cited on page 57.)

[167] Daniel S. Roche, Daniel Apon, Seung Geol Choi, and Arkady Yerukhimovich. POPE: Partial Order Preserving Encoding. In *Proc. Conference on Computer and Communications Security (CCS'16)*, pages 1131–1142. ACM, 2016. (Cited on page 64.)

[168] Daniel S. Roche, Adam Aviv, and Seung Geol Choi. A Practical Oblivious Map Data Structure with Secure Deletion and History Independence. In *Proc. IEEE Symposium on Security & Privacy (S&P'16)*. IEEE Press, 2016. (Cited on page 21.)

[169] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à Voter: A Voter-verifiable Voting System. *Transactions on Information Forensics and Security*, 4(4):662–673, December 2009. (Cited on page 112.)

[170] Cetin Sahin, Victor Zakhary, Amr El Abbadi, Huijia (Rachel) Lin, and Stefano Tessaro. TaoStore: Overcoming Asynchronicity in Oblivious Data Storage. In *Proc. IEEE Symposium on Security & Privacy (S&P'16)*. IEEE Press, 2016. (Cited on pages 4, 11, 12, 32, 33, 53, 57, 58, 59, 115, and 152.)

[171] Stefan Schiffner, Sebastian Clauß, and Sandra Steinbrecher. Privacy, Liveliness and Fairness for Reputation. In *Proc. Current Trends in Theory and Practice of Computer Science (SOFSEM'11)*, volume 6543 of *LNCS*, pages 506–519. Springer Verlag, 2011. (Cited on pages 111 and 112.)

[172] Michael Schläpfer, Rolf Haenni, Reto Koenig, and Oliver Spycher. Efficient Vote Authorization in Coercion-resistant Internet Voting. In *Proc. Conference on E-Voting and Identity (VoteID'11)*, LNCS, pages 71–88. Springer Verlag, 2012. (Cited on page 112.)

[173] Claus P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Proc. Advances in Cryptology (CRYPTO'89)*, LNCS, pages 239–252. Springer Verlag, 1989. (Cited on pages 50, 108, 131, 132, and 183.)

[174] Dominique Schröder and Heike Schröder. Verifiable Data Streaming. In *Proc. Conference on Computer and Communications Security (CCS'12)*, pages 953–964. ACM, 2012. (Cited on page 60.)

[175] Dominique Schröder and Mark Simkin. VeriStream - A Framework for Verifiable Data Streaming. In *Proc. Financial Cryptography and Data Security (FC'15)*. Springer Verlag, 2015. (Cited on page 60.)

[176] Thomas Schwarz and Ethan L. Miller. Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage, 2006. (Cited on page 60.)

[177] Hovav Shacham and Brent Waters. Compact Proofs of Retrievability. In *Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'08)*, LNCS, pages 90–107. Springer Verlag, 2008. (Cited on page 60.)

[178] Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM With $O((\log n)^3)$ Worst-Case Cost. In *Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'11)*, LNCS, pages 197–214. Springer Verlag, 2011. (Cited on pages 4, 34, and 57.)

[179] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *Proc. IEEE Symposium on Security & Privacy (S&P'00)*, pages 44–55. IEEE Press, 2000. (Cited on pages 5, 64, and 67.)

[180] Sozialgesetzbuch, 2017. Online at `https://www.gesetze-im-internet.de/`. (Cited on page 1.)

[181] Oliver Spycher, Reto Koenig, Rolf Haenni, and Michael Schläpfer. A New Approach Towards Coercion-resistant Remote e-Voting in Linear Time. In *Proc. Financial Cryptography and Data Security (FC'11)*, LNCS, pages 182–189. Springer Verlag, 2012. (Cited on page 112.)

[182] Oliver Spycher, Reto E. Koenig, Rolf Haenni, and Michael Schläpfer. Achieving Meaningful Efficiency in Coercion-Resistant, Verifiable Internet Voting. In *Proc. International Conference on Electronic Voting (EVOTE'12)*, Lecture Notes in Informatics, pages 113–125. Gesellschaft für Informatik, 2012. (Cited on page 112.)

[183] Emil Stefanov and Elaine Shi. Multi-Cloud Oblivious Storage. In *Proc. Conference on Computer and Communications Security (CCS'13)*, pages 247–258. ACM, 2013. (Cited on pages 4 and 57.)

[184] Emil Stefanov and Elaine Shi. ObliviStore: High Performance Oblivious Cloud Storage. In *Proc. IEEE Symposium on Security & Privacy (S&P'13)*, pages 253–267. IEEE Press, 2013. (Cited on pages 4, 11, 32, 57, and 59.)

[185] Emil Stefanov, Elaine Shi, and Dawn Song. Towards Practical Oblivious RAM. In *Proc. Annual Network & Distributed System Security Symposium (NDSS'12)*. Internet Society, 2012. (Cited on pages 4, 57, and 58.)

[186] Emil Stefanov, Marten van Dijk, Alina Oprea, and Ari Juels. Iris: A Scalable Cloud File System with Efficient Integrity Checks. Cryptology ePrint Archive, Report 2011/585, 2011. `http://eprint.iacr.org/`. (Cited on page 60.)

[187] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An Extremely Simple Oblivious RAM Protocol. In *Proc. Conference on Computer and Communications Security (CCS'13)*. ACM, 2013. (Cited on pages 12, 24, 33, 36, 51, 55, 56, and 115.)

[188] Isamu Teranishi, Moti Yung, and Tal Malkin. *Order-Preserving Encryption Secure Beyond One-Wayness*, pages 42–61. Springer Verlag, 2014. (Cited on page 64.)

[189] Marten van Dijk, Ari Juels, Alina Oprea, Ronald L. Rivest, Emil Stefanov, and Nikos Triandopoulos. Hourglass Schemes: How to Prove That Cloud Files Are Encrypted. In *Proc. Conference on Computer and Communications Security (CCS'12)*, pages 265–280. ACM, 2012. (Cited on page 60.)

[190] Marco Voss, Andreas Heinemann, and Max Mühlhäuser. A Privacy Preserving Reputation System for Mobile Information Dissemination Networks. In *Proc. Security and Privacy for Emerging Areas in Communication Networks (SECURECOMM'05)*, pages 171–181. IEEE Press, 2005. (Cited on pages 111 and 112.)

[191] Stefan G. Weber, Roberto Araújo, and Johannes Buchmann. On Coercion-Resistant Electronic Elections with Linear Work. In *Proc. Conference on Availability, Reliability and Security (ARES'07)*. IEEE Press, 2007. (Cited on pages 106 and 112.)

[192] Peter Williams, Radu Sion, and Bogdan Carbunar. Building Castles out of Mud: Practical Access Pattern Privacy and Correctness on Untrusted Storage. In *Proc. Conference on Computer and Communications Security (CCS'08)*, pages 139–148. ACM, 2008. (Cited on page 58.)

[193] Peter Williams, Radu Sion, and Alin Tomescu. PrivateFS: A Parallel Oblivious File System. In *Proc. Conference on Computer and Communications Security (CCS'12)*, pages 977–988. ACM, 2012. (Cited on pages 11, 58, and 59.)

[194] Khin Than Win, Willy Susilo, and Yi Mu. Personal Health Record Systems and Their Security Protection. *Journal of Medical Systems*, 30(4):309–315, 2006. (Cited on page 60.)

[195] Andrew Chi-Chih Yao. How to Generate and Exchange Secrets. In *Proc. Annual Symposium of Foundations of Computer Science (FOCS'86)*, pages 162–167. IEEE Press, 1986. (Cited on page 31.)