
ACE of AUTHENTICITY
25
THE VERSATILITY of DIGITAL SIGNATURE SCHEMES

Dissertation

Ace of Authenticity
The Versatility of Digital Signatures

Dissertation zur Erlangung des Grades der Doktorin der Naturwissenschaften
der Fakultät für Mathematik und Informatik der Universität des Saarlandes
eingereicht von Stella Lucien Wohnig

Saarbrücken, 2025

Angaben zum Kolloquium

Dekan: Prof. Dr. Jan Reineke
Datum der Verteidigung: 19. Juni 2026
Prüfungsausschuss:

- **Vorsitz:** Prof. Dr. Martina Maggio
- **Gutachter:** Prof. Dr. Markus Bläser, Dr. Geoffroy Couteau, Prof. Dr. Antoine Joux
- **Akademischer Beisitzer:** Dr. Felix Rohrbach

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, 2025

gez. Stella Lucien Wohnig

"Find solace in the privilege to pursue;
Most people are crushed into servitude."

- Jonathan Bree

Zusammenfassung

Digitale Signaturen sind ein Eckpfeiler moderner Kryptographie und ein vielfältiger Baustein. Diese Arbeit untersucht ihre Anwendungsfelder und ergänzt neue Konzepte und Konstruktionen zur Nutzung in adversariellen, verteilten Systemen.

Ein erster Beitrag ist die Studie *universeller Ring-Signaturen* (URS). Im Kontext von Whistleblowing wurden Ring-Signaturen (RS) eingeführt, die anonyme Attestierung einer Sendergruppen-Zugehörigkeit erlauben. Während bisherige RS Senderschlüssel kompatibler Signaturschemata erfordern, erlauben URS die Kombination beliebiger Schlüssel und ermöglichen anonymes Whistleblowing in unkooperativen Umfeldern. Unsere Standardmodell-Konstruktionen basieren auf superpolynomiellen Standardannahmen, Zeugenverschlüsselung oder Ununterscheidbarkeits-Obfuskation (iO).

Zweitens führen wir *signaturbasierte Zeugenverschlüsselung* (SWE) ein, bei der Entschlüsselung eines Chiffrats durch eine Mindestzahl von Signaturen ausgewählter Parteien möglich wird. Auf Basis bilinearer Abbildungen entwickeln wir effizientes SWE für BLS-Multisignaturen im Zufallsorakel-Modell. Wir integrieren SWE mit einer Blockchain, um effiziente zeitbasierte Verschlüsselung ohne vertrauenswürdige Initialisierung zu realisieren.

Wir reduzieren zudem die asymptotische Komplexität von SWE und stellen im Standardmodell basierend auf iO und punkturierbaren Signaturen das erste *kompakte* SWE-Verfahren vor, dessen Chiffratgröße sublinear in der Zahl Signierender wächst.

Abstract

Digital signatures are a cornerstone of modern cryptography and a versatile building block. This thesis explores their versatility and contributes novel concepts and constructions for their use in adversarial, distributed settings.

Firstly, we introduce the study of *universal ring signatures* (URS). Ring signatures (RS), introduced to enable whistleblowing, allow to anonymously certify that a message originates from some member of a signer group. While previous RSs rely on signers' keys to belong to compatible signature schemes, a URS works for arbitrary key combinations from any combination of potentially different signature schemes, enabling whistleblowing in uncooperative settings. Our standard model constructions rely on superpolynomial standard assumptions, witness encryption, or indistinguishability obfuscation (iO).

Secondly, we introduce *signature-based witness encryption*, where decryption requires a threshold number of signatures from selected parties. Based on bilinear maps, we develop efficient SWE for BLS multi-signatures in the random oracle model. We integrate SWE with a blockchain to realize efficient time-release encryption without a trusted setup.

Thirdly, we reduce the asymptotic complexity of SWE and present the first *compact SWE* with ciphertext size growing sublinearly in the number of signers. This standard model construction is based on iO and puncturable signatures.

Contents

Acknowledgements

It is with an odd feeling that I sit here at the end of a five-year-long journey and the beginning of another. Doing a PhD was nothing like what I had ever imagined it to be, for better or worse, and I am eternally grateful for the connections I have made on the way, the guidance, support, and inspiration. This will be a rather emotional trip down memory lane on how I came to be in the position to submit this manuscript and I will try to mention everybody deserving an acknowledgement, but if you dear reader do not find yourself, know that this was written on a night with an energy drink and that you are meant as well!

Privilege. First, I want to acknowledge the great privilege it has been to grow up and live in a country and at a time, where academia, at least in our field, offers living wages to young researchers. Growing up in a common family would not have qualified me to take part in science, when it was a pastime for the elites. Being born a *girl*, enrolling into a university was impossible in Germany until 1900.

As I grew up in the countryside, I have heard "girls can't do that" way too many times. And I want you, dear reader, to know: Yes, we absolutely can.

Fostering Inspiration. Thankfully, I had ample support in my life. I thank, first of all, my parents, who always tried their best to keep up with me asking "why" and allowed me to (controlledly!) set way too many things on fire and run chemistry experiments in the barn. Secondly, I thank the *good* teachers that I had, that inspired critical thinking and discussions. A special mention goes to Matthias Bergmann, who offered an excellent extracurricular math class, and who supported us to do as many math competitions as we could – offering not only a direction for me to throw the "why" in my brain at, but also a welcoming social circle of nerds. I remember him saying I'd be a professor some day, and me thinking I'd study game design at the time – but in the end I think this early connection with math ended up making me pursue a degree in math.

I also had the privilege of receiving a scholarship for my studies and was able to take more lectures than I otherwise would have. I am grateful for many good lecturers during the years, including Dr. Yasmine Sanderson, who got me interested in science outreach and algebra, Prof. Wolfgang Ruppert, who introduced me to cryptography and was a very involved advisor for my Bachelor's thesis, Prof. Rolf Wanka, who got me into complexity theory and inspired me to pursue theoretical informatics, and Dr. Nico Döttling, who introduced me to *modern* cryptography and some day offered me to do a PhD with him.

Support at CISPA. I thank Dr. Nico Döttling for this offer, and his involvement and inspiration during the initial part of my PhD journey. I further thank Prof. Markus Bläser, who took me on as a student later in my PhD and who has provided me with structure and guidance as my supervisor for this thesis, and Dr. Wouter Lueks, who has reminded me, as often as necessary, in an empathetic, but very pointed way that I should be less of a perfectionist and really just hand the thesis in at some point. Further, I thank David Pfaff from CISPA and Dr. Michelle Carnell from the UdS Graduate School, as without their assistance I may not have been able to finish my PhD here in Saarbrücken.

My Doctoral Siblings. The CISPA has been a very inspiring workplace and I loved the welcoming environment among my fellow PhD students. From whiteboard

discussions, chit-chat over coffee breaks, joint conference trips to gaming nights, picnics, and birthday invites, I have found not only coworkers, but real friends. While many people have been amazing, I would like to specifically mention: Anne, who organized many Christmas gatherings with me and got me interested in quantum computation after she fell madly in love with it. She always has a smile, amazing home-made food and the coolest vacation pictures with her – I'm happy I got to stay a few nights in Trondheim with her and our colleague Jesko! Chuanwei, who has become my little brother, always ready for a little bit of chit-chat and queer shenanigans, but also always ready to body-double, discuss or proof-read when I needed help with conference papers and ideas. He's also been dragged across a few too many cities by me, despite him always just wanting to chill and have a warm meal. And then, there's the Italian trifecta: Giacomo, who miraculously does not use a calendar, but somehow is the one to organize PhD gatherings. He's also a beaming ball of positive energy, shares my enthusiasm for random blockchain papers, and helped with proof-reading parts of this thesis. Riccardo, who keeps things together, makes the best quiche, and always has cool anime recommendations. He's also great at making me feel like I should do more sports, which is a good thing, really. Eugenio, the style-icon, who discussed some ring signature ideas with me, that never made it to papership, but who has also taken me to the best party(s) in Saarbrücken.

Growing Up in The Scientific Community. I was especially privileged and got to travel to many conferences and workshops. The first conference was TCC 2019 in Nuremberg and I still vividly remember how there was one single talk that I was able to understand at the time and how anxious I felt to talk to all the "grown up" researchers. But when I broke out of my shell, I felt very welcome and understood in the scientific community. I remember getting so much good advise over the years.

Since then, I've been able to visit *eleven* countries (and some more in private) during my time as a PhD student and I got to meet countless cool and inspiring people. I will try to recall some of the best moments for your enjoyment.

Firstly, I got to stay for a whole month in Manchester visiting Dr. Bernardo Magri during our work on compact SWE (Spoiler: It's in Chapter 5!) This was during one of the low points of my thesis, where I seriously considered giving up, and I appreciated the guidance and relaxed atmosphere in Bernardo's group, which really contributed to me finding new inspiration to see things through. (Did you know his PhD thesis has an *amazing* artwork cover?) What really made my stay outstanding was his student Chris, who welcomed me super warmly, introduced me to his friends, took me on a trip to Liverpool and to all the best food places in the city. He even took me to a "haunted" hotel, after suffering from my constant oversharing about antique houses.

Another amazing person I met on this project was Gennaro, who I last saw in Madrid eating pizza, while he was super stressed and sleep deprived from organizing Eurocrypt. He has been super fun to work with and throw ideas at, but he also had an open ear when my PhD experience was tough and I'm so grateful for the support.

My favourite travel companion has been Yingfei, who I met at Asiacrypt 2022 in Taipei after being introduced by our friend Chuanwei. She helped me out so much with Chinese in Taiwan, let me take her to super random places with amazing views, booked our trains and made me eat *every imaginable kind of seafood*. I hate seafood. But I was very German and made her *walk without warm lunch*, so we are even. It was super amazing to meet her again in France and be able to return the favour a tiny bit with my subpar French. She's organized the most rare instant ramen from China for me, as well.

Another good friend I made in the scientific community has been Akin, who shares my taste in music and explored the Vienna and Madrid (goth) nightlife with me and (unwillingly) became part of a side quest to obtain rare energy drinks from Denmark. He also helped me in navigating some of the frustrations of a PhD, was there when I needed help, and proof-read parts of this thesis with astonishing patience. I'm super grateful for all the support.

Last year, Akin invited me to join his project working on CVRFs (Spoiler: This one didn't fit into my thesis, but a short write-up is in § 1.4). I got to visit Vienna for this project and met Prof. Krzysztof Pietrzak and his cool group including Ben, Charlotte, Christoph, and Miguel. It was super lovely to have a picnic with the group at Donauinsel in the 30-something degree summer heat and melt in meeting rooms and cafes working.

Some other mentions in no particular order: I thank Rahul for the coolest Rump Session talk, Lisa for an amazing mentoring session, Aron for dinner in Jerusalem and showing me arround when I was in Århus, Anca and Paola for always knowing where to go dancing, Anasuya for being bubbly company in Darmstadt, Chris for queer representation, Joël for checking in on me and being part of my random sidequest networking two masters' students, the other Bernardo for encouraging many social outings and randomly wandering Madrid at night, Marcel for the most random party-invite and for being one of the first researchers to bond with me at TCC, Khanh for Karaoke and forcing me to do my first Rump Session talk, Hunter for joining me in cozy campfire and puzzle events, Hans for being all vibes and making super cool music, Jelle for being an inspiration and then disappearing, being even more of an inspiration thereafter. And generally, I thank the community for often being so random and so inspiring!

Coming back to the topic of growing up: Perhaps one of the greatest learnings I have had is that "grown ups" usually also have no clear idea what they are doing, they just get better at winging it. And so, when I started to understand more and more talks at conferences and giving advice to more junior PhD students, even though I still don't *know*, I suppose I am a grown up now.

Non-Academic Friends. In this PhD journey that has been constant insanity, inspiration, and burnout, I have been blessed to have non- (and post-)academic friends at my side, who made life in *Saarbrücken* of all places much more bearable!

I thank all the friends I have made travelling and the kindness that humans have extended to me.

I thank Nyx for her calming nature, guidance and making me reflect.

I thank Atul for being my art friend, dragging me out of the house to paint, go to Jam sessions, but also being a philosophy friend for deep talk.

I thank Merwyn for the autumn walks and the music.

I thank Desi for the shopping trips, coffee meet-ups, always showing up to my silly motto partys and her directness and anger, when I'm still people-pleasing.

I thank Thunis for existing and in particular Alex, Clara, Sophie, Max, and Sarina for the joy I experienced playing in a theater piece with them.

I thank Nicole for all her rants against working in academia and all the times she's shown me random stuff in Leipzig.

I thank Arkadius and Vas for always having a soft couch and a warm meal, hosting parties and providing a different perspective to my problems many times.

I thank Sascha for his calm nature, despite always being stressed. I thank him for his guidance, the way he always shows up in an emergency, the amazing food, the most obscure life stories, the campfire-making.

I thank Roman for being one of the people who have been in my life the longest, the quiet knowing, the infrequent visits in different cities (it is me, I never have time), cozy Christmas celebrations, checking in on me, and always having an open ear and arms.

I thank Daniel for being my best (male) friend[1] who listens to all of my life's complaints with patience, who lures me into new creative projects, into the woods, into new experiences, who is the best at bringing up the most random metaphysical questions at 3 am, who completes the part of my adult functioning, where I have to *buy things*, like he talked me into buying so much furniture until my house became warm and cozy and he always ate my cookies in my baking phase. Daniel feels like a part of myself without which I would simply not be complete. One day we will make an amazing music album, if we don't both have too much random other side quests. I also thank his girlfriend Caro for accepting me as their house cat and feeding me with cheesy pasta. Caro is the amazing artist that made my original sketches for my thesis cover come to life!

Last, but not least, I thank Flora for being my best (non-male) friend. She invested a metric ton of unpaid labor, without which this thesis (and many other things in my life) would have been near impossible. From support with fixing my car, fixing my flat, fixing my calender, fixing my hair to late night therapy sessions, body-doubling, proof-reading, encouraging me to be my best self and encouraging me to allow myself to be my lazy self. I thank her for the spontaneous travel adventures, party adventures, theater adventures, coming to dance class with me and starting to be a better lead than me already, being my cheerleader when I choose to do *too many things again* and get overwhelmed with life. Showing up the night before a conference to listen to me stress over my slides while packing my suitcase. She's the best communicator and empath I know and I'm still trying to learn her art of effortless diplomacy. But she's also really honest and blunt, when I need to be reminded that *I did this to myself* so I can grow (and not accept two back to back conference invites next time – maybe). She makes the world around her a more social, unbroken and pretty place and I'm grateful for having her tag along my journey.

Thanks for Reading. It is a fact that academic literature is written and consumed mostly by academics and I have probably spent way more time reading my own thesis than all future readers combined ever will. Therefore, if you are a reader of this thesis, thanks for taking the time.

[1] Im Deutschen ist das immerhin schön sortiert mit bestem Freund und bester Freundin, aber wir müssen das hier mit dem besten auch nicht päpstlicher als der Papst nehmen.

Chapter 1

Introduction

Trust is a necessary condition for communication, but in digital settings, it cannot simply be assumed. Where trust in the real world is interpersonal, in the digital space, trust is carefully constructed in an interplay of protocols and cryptographic means.

One of the core guarantees necessary in online communications is the *authenticity* of messages – namely, we want to ensure that the people and services we talk to are who we think they are. This already becomes apparent in everyday settings like group messaging, but in a society that increasingly moves infrastructure, commerce, and decision-making into the digital sphere, trust in our digital communications has become a ubiquitous but often silent requirement.

We do not check by hand whether we are connected to the correct server when logging into an online service and sharing private information, but rather, we *trust our browser* to alert us to an insecure connection.

The unsung hero of authentication processes is, more often than not, a *digital signature scheme*. A user of a digital signature scheme produces a public verification key vk alongside a signing key sk, allowing them to publicly distribute vk. When sending a message, they use sk to produce a signature that anyone can verify using vk to confirm it was indeed created by this user for that message. The idea of digital signatures dates back to the seminal proposal of Diffie and Hellman in 1976 [DH76], which envisioned public-key cryptography as a means to secure open networks, a concept that has now become a part of everyday life.

The ease with which we navigate the digital world is enabled behind the scenes by vast and layered *public-key infrastructures* (PKIs), providing hierarchical assurances of trust. When we connect to a website, the browser verifies that its credibility is backed by a valid certificate chain, which authenticates the site's identity via a path of digital signatures up to a trusted root CA. This certificate chain ties the website's identity to public key material, which can be used to establish a secure, encrypted communication channel with the underlying server using TLS. Certificates are used in messaging apps to authenticate users and establish end-to-end encrypted channels between them. When your operating system performs software updates, it checks for valid vendor signatures. Critical emails and documents may be signed. Indeed, the very notion of "trust" in the digital sphere is often synonymous with the presence of a valid signature.

Despite authenticity being the guarantee most commonly associated with signatures, many variants and use cases of signatures exist that go far beyond this property. In this work, we want to shine a light on a fraction of the many possibilities that make signatures one of the most versatile and most widely deployed cryptographic schemes today.

1.1 Historic View on Digital Signature Schemes

Foundations. To set the stage, let us recall the origins of digital signature schemes and retrace some of the important historical developments in the field. After Diffie and Hellman [DH76] laid the groundwork for this area of study in 1976 by conceptualizing digital signatures and providing a blueprint of how they could be instantiated from trapdoor functions, it took only two years for the first practical instantiation of digital signatures to be proposed: Rivest, Shamir, and Adleman presented in [RSA78] the RSA public key encryption scheme and its accompanying RSA signature scheme, whose security relies on the presumed hardness of factoring large integers. The RSA signature scheme is compellingly simple and efficient to implement and still enjoys widespread use today; as of September 2025, roughly 75% of current PKI certificates employ RSA signatures as reported by Cloudflare [Clo25]. Another early alternative that remains influential today is the ElGamal signature scheme [ElG85], which is based on the discrete logarithm problem and served as the basis for the Digital Signature Algorithm.

While these early signature proposals are still relevant today, it is important to note that their original proposal predated the formal cryptographic definition of security for signature schemes that we consider standard now. In 1988, Goldwasser, Micali, and Rivest [GMR88] introduced the notion of *existential unforgeability under chosen message attacks* (EUF-CMA), which continues to be the security notion of choice for modern signature schemes. To fulfill this notion, an efficient attacker, who is given a public verification key vk and access to an oracle that provides it with valid signatures under vk on messages of its choice, should be unable (except with negligible probability) to provide a valid forgery, consisting of a valid signature under vk on any message not previously queried to the signing oracle. In [GMR88], it is discussed that previous signature schemes, including the early textbook variants of RSA and ElGamal, fail to achieve EUF-CMA security. Over time, however, both schemes were modified to meet this definition.

Introducing Hashing. A decisive step in this evolution was the introduction of the Fiat-Shamir transform [FS87], which is applied to an interactive identification protocol, where a prover needs to prove their identity by correctly responding to random challenges given by a verifier. This notion of an identification protocol was inspired by advances in zero-knowledge proof systems due to Goldwasser, Micali, and Rackoff [GMR85], but the term was coined in [FS87]. The transform turns such a scheme into a non-interactive one, effectively yielding a digital signature scheme. This is done by replacing the random challenges the prover would receive from the verifier by computing a hash of the prover's own transcript.

This has led to a shift from ad hoc algebraic constructions, such as the early RSA or ElGamal proposals, to a streamlined paradigm of providing an interactive identification scheme, applying the Fiat-Shamir transform, and receiving a signature scheme. A canonical example of this approach is the Schnorr signature [Sch90], as Schnorr first introduces an identification scheme based on the discrete logarithm (based on such a scheme presented by Chaum, Evertse, and van de Graaf in [CEG88]) and then derives signatures by applying the Fiat-Shamir transform. The resulting scheme is another discrete-logarithm-based scheme that is conceptually similar to ElGamal signatures, but yields much smaller signature sizes.

The key to achieving provable security for signatures based on the Fiat-Shamir transform was the introduction of the *random oracle model* (ROM) by Bellare and Rogaway [BR93], which idealizes cryptographic hash functions as truly random functions that are revealed only through oracle queries. In the case of RSA, EUF-CMA

security is achieved in the ROM for variants that sign padded and hashed messages. Specifically, Bellare and Rogaway present both RSA-FDH [BR93], which requires a full-domain hash and employs a hash-then-sign approach, and the more practical RSA-PSS [BR96], which introduces randomized padding to allow for the use of standard cryptographic hash functions. For discrete-logarithm-based schemes, Pointcheval and Stern [PS96] showed that Schnorr signatures and hashed versions of ElGamal signatures are EUF-CMA secure in the ROM.

Standardization. The practical importance of digital signatures was soon recognized at a standardization level, marking an early point of real-world adoption: In the early 1990s, the U.S. National Institute of Standards and Technology (NIST) published the Digital Signature Standard (DSS), in which they initially standardized the "Digital Signature Algorithm" (DSA), essentially a variant of ElGamal and Schnorr signatures.[1] In the current version of the DSS [Nat23], the standardized signatures are RSA and elliptic-curve variants of DSA, specifically ECDSA [JMV01] and EdDSA [BDLSY12]. Regular (finite-field) DSA has been deprecated due to its need for bigger key sizes and signatures for similar security levels as compared to its elliptic-curve counterparts.

Looking forward, it is important to note that the schemes standardized in [Nat23] are all based on the hardness of integer factorization or discrete logarithms in elliptic-curve groups. These assumptions, however, are rendered insecure against adversaries running large-scale quantum computers, as Shor's algorithm [Sho94] solves both problems in polynomial time. While such quantum computers may not be available yet, the impending vulnerability has led to an international effort towards the standardization of *post-quantum* signature schemes.

In particular, NIST initiated a competition in 2016 [Nat] with the aim of standardizing different types of post-quantum public-key cryptography. Regarding signature schemes, the lattice-based scheme CRYSTALS-Dilithium [Duc+18] and the hash-based scheme SPHINCS$^+$ [Ber+19] were standardized in 2024 in NIST publications [Nat24a; Nat24b]. Furthermore, the lattice-based FALCON [Pre+22] was selected for standardization. New candidates are also continually being considered to diversify the assumptions on which we base post-quantum signatures. A shift towards these schemes can be expected in real-world usage in the future.

1.2 Versatility of Digital Signatures

The modern digital landscape is increasingly decentralized, and the need for authenticity in digital systems often exists in the interplay with other guarantees such as anonymity or confidentiality.

While the canonical definition of digital signatures presented above ensures the authenticity of messages in protocols where a single signer needs to convince a receiver of their identity, the primitive has evolved over time to be remarkably versatile. Many variants of signatures have been studied, with some focusing on extending the notion of authenticity to distributed settings and others incorporating additional functionality and security goals beyond authentication.

1.2.1 Distributed Signatures

As we have previously outlined, trust in the digital world is often synonymous with signatures. While the single-signer notion of signature schemes discussed thus far

[1] The reason that the efficient Schnorr scheme was not standardized is commonly believed to be an active patent held by Schnorr at the time.

is very powerful, as evidenced by its role in enabling authenticated communication on the internet via hierarchical PKI, a natural goal for researchers is to capture the nuances of distributed trust in a decentralized digital world.

Threshold Signatures. In 1987, Desmedt [Des88] posed multiple questions on how to construct cryptographic schemes that account for groups of receivers or senders. Specifically, he asked how we can distribute the ability to sign among multiple parties. This idea evolved into what we now call *threshold signatures*: a secret signing key is divided among n parties such that only subsets of size at least t can collaborate to produce a valid signature under a shared verification key. This directly addresses scenarios with distributed trust, as no single entity holds the signing key in its entirety, but a qualified subset can sign. For instance, in corporate settings, it might be desirable to require multiple parties to sign off on a contract before it goes into effect.

The first threshold signature was given by Desmedt and Frankel in [DF92] for RSA signatures, but many subsequent instantiations have been proposed. For example, threshold Schnorr schemes exist from a line of work on distributed key generation due to Pedersen [Ped91] and Gennaro, Jarecki, Krawczyk, and Rabin [GJKR99], and a very efficient threshold ECDSA was suggested by Gennaro and Goldfeder in [GG18] and has been used in practice for securing Bitcoin wallet keys by offering custodial wallet services.

Multi-Signatures and Aggregate Signatures. Before the formal notion of threshold signatures was developed, Itakura and Nakamura [IN83] proposed one of the earliest forms of collective signing, introducing what they called *multi-signatures*. In their construction, multiple users cooperate to produce a single joint signature on a common message, which can be verified against all of their public keys. The goal was collective authorization – ensuring that a document is endorsed by all intended signers – rather than distributing trust among a signer set.

Modern incarnations of this notion focus on efficiency as they allow for compressing multiple signers' independently generated signatures on the same message into a single signature verifying for that set of signers. In contrast, *aggregate signatures*, introduced by Boneh, Gentry, Lynn, and Shacham [BGLS03], extend this concept by allowing signatures on different messages to be compressed into a single aggregate signature.

In their original proposal of this notion, a construction is given based on BLS signatures due to Boneh, Lynn, and Shacham [BLS01]. These signatures rely on bilinear pairings and provide elegant algebraic homomorphism properties. The construction in [BGLS03] required signatures on *distinct* messages, but this restriction is somewhat artificially introduced for technical reasons. The authors themselves suggest that it can be overcome by prepending users' verification keys to the message before signing. Alternatively, the plain BLS aggregation scheme can be shown to be secure when used as a multi-signature in a key-registration setting. This is shown under the strong assumption that all secret keys are known by Boldyreva [Bol03], or under the weaker assumption that signers provide proofs of possession of their keys by Ristenpart and Yilek [RY07]).

BLS signatures are commonly used in blockchain applications as they yield especially compact aggregate signatures. BLS signatures are employed, for example, in Ethereum or Chia to drastically reduce bandwidth and verification overhead, making large-scale consensus protocols practical. In Bitcoin, Schnorr-based multi-signatures are used, which date back to an early construction due to Bellare and Neven [BN06] and have been optimized for efficient blockchain use (compare [BDN18; MPSW19]).

1.2.2 Privacy-Preserving Signatures

Two security goals that cryptographers have integrated with signatures early on are *privacy* and *anonymity*. At first glance, these appear to be in tension with authenticity, which aims to bind a message to an identity. However, combining the authenticity of a signature with message privacy or signer anonymity opens up new use cases, as we will elaborate.

Blind Signatures. In the seminal work of Chaum in 1982 [Cha82], he introduced the notion of *blind signatures*. In such a scheme, a user can obtain a valid signature on a message from a signer without the signer learning the message itself. The real-world analogue that he likens this scheme to is signing a message inside a charcoal-lined envelope. Signing happens without visibility of the content, but the signed document can later be retrieved by opening the envelope.

As an application, Chaum suggests that this scheme can be used towards establishing anonymous digital cash. Banks could issue digital coins by blindly signing user-chosen serial numbers, ensuring both unforgeable currency and payer anonymity.

This line of work has evolved into a broad research area and inspired anonymous credential systems, introduced conceptually by Chaum [Cha90] and realized efficiently in later works such as those of Camenisch and Lysanskaya [CL01]. These schemes allow users to obtain credentials attesting to some attributes (e.g. being of voting age, residence in a certain country), which can be presented to services while revealing only the relevant attributes and staying pseudonymous as opposed to showing an ID card which discloses unnecessary personal information. An interesting sub-category is that of anonymous tokens such as Privacy Pass [DGSTV18], introduced by Davidson et al., and deployed today by major web providers. Privacy Pass allows users to obtain blindly signed tokens when solving CAPTCHA challenges, which can later be redeemed anonymously to bypass further challenges, resulting in a better user experience without linkability across sessions.

Group Signatures. While blind signatures protect the privacy of the *message*, another line of work has aimed at protecting the identity of the *signer*. This ambition was first formalized in the notion of *group signatures* by Chaum and van Heyst [CH91]. In a group signature scheme, any member of a group can sign a message on behalf of the group. The resulting signature can be publicly verified to have originated from one of the group members, without publicly revealing the identity of the signer, providing anonymity within the group.

Crucially, group signatures are designed with cooperating groups in mind – they typically rely on a trusted setup for members' key material, and accountability is added by having an *opening authority*, which can reveal signer identity in case of disputes. This makes them suitable for corporate settings, where employees sign contracts on behalf of a firm.

Ring Signatures. In contrast, *ring signatures*, introduced by Rivest, Shamir, and Tauman in 2001 [RST01], remove the need for any group setup or trusted authority. Instead, a signer can form an ad hoc ring composed of public verification keys and produce a signature that proves only that the message was signed by one of them, without revealing the signer's identity. Crucially, a ring signature can be made without any cooperation or consent of other ring members.

This notion targets decentralized or adversarial environments, and the authors of [RST01] envisioned its primary application to be whistleblowing, where trust is minimal and privacy must be preserved even against in-group members.

Ring signatures have become a standard tool for anonymous attestation, with constructions developed for many widely-used signature schemes. While the original [RST01] was instantiated over RSA keys, ring signatures have, for instance, been proposed for BLS by Boneh, Gentry, Lynn, and Shacham [BGLS03], and a generic construction was given by Abe, Ohkubo, and Suzuki in [AOS02] for Fiat-Shamir style signatures, such as Schnorr.

Further, several extensions of ring signatures have been proposed to make them suitable for different use cases. One notable variant is that of *linkable ring signatures*, introduced by Liu, Wei, and Wong in [LWW04], which allow public detection of whether two ring signatures were issued by the same signer, while otherwise maintaining anonymity. Use cases encompass the prevention of double voting in anonymous electronic voting systems and double spending in anonymous e-cash. Notably, the Monero blockchain employs a variation of linkable ring signatures introduced by Goodell, Noether, and Blue [GNB19] to conceal from which account a transaction's funds originate.

1.2.3 Confidentiality and Conditional Access

Another central notion in cryptography is *confidentiality*, which is typically associated with encryption but can also be combined with signatures. In such settings, either the signatures or even the underlying signed messages themselves are only available to targeted recipients or under certain conditions. We will introduce some of these notions and related concepts.

Signcryption. A straightforward question is how to achieve confidentiality and authenticity for a message sent in one go. This can be achieved in a natural way by combining signing and encrypting the signature alongside the message. As an alternative, Zheng suggested *Signcryption* [Zhe97], which focuses on integrating signing and encrypting into one operation that enjoys lower computation and communication cost than the sign-then-encrypt approach. This is useful in bandwidth-constrained environments such as secure messaging or the Internet of Things.

Verifiably Encrypted Signatures. Another notion that targets the confidentiality of a signature rather than the underlying message is a *verifiably encrypted signature*. The term was coined by Boneh, Gentry, Lynn, and Shacham [BGLS03], but the notion dates back to work on fair exchange of signatures by Asokan, Shoup, and Waidner [ASW98]. In such a scheme, a signature is only published in encrypted form, accompanied by a proof that it decrypts to a valid signature on a given message under a known verification key. This is used in optimistic fair exchange settings like fair contract signing. In the presence of an intermediary, two parties send each other verifiably encrypted signatures that the intermediary *could* reveal. Afterwards, they exchange the actual signatures. Neither party has to risk exposing its signature first, and the intermediary only needs to step in if one party aborts, ensuring fairness without being otherwise involved in the exchange.

Timed Signatures. Another primitive that was motivated by fair exchange of signatures is that of *timed signatures* as proposed by Boneh and Naor [BN00]. Building on the idea of time-release encryption, which was pioneered by Rivest, Shamir, and Wagner [RSW96], the notion of timed signatures allows a signer to publish a commitment to a signature. Later, the signer can either open this commitment by releasing the signature, or the receiver can execute a forced opening phase to retrieve the signature independently. The security guarantee is that forced opening takes a

predetermined time, while no information is leaked to the receiver before that time has passed. Intuitively, this ensures that a signature, if not voluntarily disclosed by its signer, remains hidden until a specified time has elapsed, at which point it inevitably becomes available.

1.3 Our Contributions

After we have seen how digital signatures have evolved into a remarkably versatile primitive, we want to now give some high-level motivation and an overview of our contributions in the further study of possibilities provided by digital signature schemes.

In this introduction, we provide a brief overview of the directions covered in this thesis. We will begin each chapter with a more formal list of our respective contributions, including theorem statements and individual authors' contributions, before delving into each chapter's technical overview and comparison to related work.

1.3.1 Anonymous Whistleblowing in Adversarial Settings

As we have previously identified, there are a plethora of ring signature schemes available for use today. These are based on a wide variety of signature schemes such as RSA, Schnorr, or BLS signatures, and each scheme comes with its own structure, assumptions, and intended applications. The fact that many deployed signature schemes support compatible ring signature constructions reinforces the idea that anonymous certification is broadly achievable – at least in principle.

The real-world deployment of ring signatures in Monero demonstrates the practicality of such schemes, albeit in a setting where users actually *want* to give each other anonymity, as they mutually benefit from the anonymous transaction ledger. In such a setting, users will naturally adopt a common signature scheme, and an efficient ring signature based on that scheme is sufficient to achieve anonymous attestations.

However, the question we ask is: what happens to ring signing capabilities in a setting where users object to being used in an anonymity ring?

We consider again the whistleblowing scenario discussed by Rivest, Shamir, and Tauman [RST01]: A government official wishes to anonymously leak sensitive information to a journalist, while convincing them that the message originated from someone within an authorized department. In such a setting, ring signing can be seen as an *adversarial act* against the other members of the department.

Crucially, such a whistleblower cannot assume that other members of the department are willing to cooperate. In the modern world, with its plethora of signature schemes, the other officials to be included in an anonymity ring may have chosen public keys from vastly different signature schemes, and we cannot assume that they had ring signatures in mind when choosing their keys. From the perspective of a government agency, it would even make sense to mandate that employees refrain from using digital signature schemes that lend themselves to ring signing.

In such a setting, the unavailability of compatible ring signing schemes for even one underlying signature scheme can allow to prevent leaks via this anonymity tool. Moreover, even in more benign settings, an adoption paradox can arise: the very knowledge that a signing scheme supports ring signing may deter users from adoption, as they might want to avoid being framed.

This raises a natural question: can users protect themselves from being used in an anonymity ring?

Universal Ring Signatures. We answer this question negatively in our paper [BDW22], a joint work with Nico Döttling and Pedro Branco, and introduce the

study of *Universal Ring Signatures* (URS). A URS scheme allows a user to construct a ring signature over any set of public verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell$, without requiring these keys to originate from a common signature scheme or making any structural assumptions on the keys or their associated signing schemes.

We note that prior works such as Abe, Ohkubo, and Suzuki [AOS02] and more recently Goel, Green, Hall-Andersen, and Kaptchuk [GGHK22] also sought to generalize ring signatures beyond a single underlying scheme, focusing on Fiat-Shamir style signatures. These works target efficient ring signatures for many commonly deployed schemes, but still require some structure and rely on the random oracle model, prioritizing efficiency over full universality. We refer to our related work section § 3.2.4 for more details.

In Chapter 3, we present our findings from [BDW22] by introducing a formal definition of our new notion of universal ring signatures and providing several constructions of URS in the standard model – that is, without use of the ROM or a trusted setup:

- We construct a URS scheme based on signature schemes with superpolynomial security via complexity leveraging. The resulting signature is compact in the sense that its size grows only logarithmically in the size of the ring.
- We present a non-compact URS construction under standard polynomial-time assumptions, using witness encryption.
- Finally, we show how to achieve compact URS under polynomial assumptions, albeit relying on the potentially even stronger tool of indistinguishability obfuscation.

These results demonstrate that universal ring signatures are not only a natural extension of existing ring signature notions but also achievable under a range of cryptographic assumptions. We establish that ring signing is possible, albeit not concretely efficiently, even in maximally non-cooperative settings.

1.3.2 Conditional Release in Signature Ecosystems

The second line of work introduced in this thesis explores new ways of combining the trust and authenticity provided by signatures with confidentiality. Specifically, we investigate how signatures themselves can serve as the mechanism that enables controlled access to encrypted information. In contrast to verifiably encrypted or timed signatures, our goal is not to use encryption to protect signatures, but rather to use signatures as a trigger that allows decryption.

The notion we are targeting is *conditional release of information*, where data is encrypted to remain hidden until some condition is met. This is a common setting in cryptography with examples of meaningful conditions ranging from data being released only when 1) a judge authorizes disclosure, 2) a specified amount of time has passed (as in sealed-bid auctions) or 3) a real-world event has occurred (such as winning a bet or the activation of a dead-man switch).

We observe that many processes that happen in the digital world will already be recorded and attested with a signature. In this vein, we want to build a conditional-release encryption in which the availability of signatures directly serves as a decryption condition.

Signature Witness Encryption. With this goal in mind, in our paper [DHMW23], a joint work with Nico Döttling, Lucjan Hanzlik, and Bernardo Magri, we introduce the notion of *Signature Witness Encryption* (SWE), in which a message m is encrypted with respect to a reference message T and a set of verification keys

$(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$. Decryption is possible if and only if signatures on T under a threshold number t of verification keys vk_i are available. In case such a threshold number of signatures is not available to an adversary, ciphertext-indistinguishability should hold for our SWE ciphertexts.

Comparison to Similar Notions. This can be likened to the notion of *Witness Encryption*, introduced by Garg, Gentry, Sahai, and Waters [GGSW13], where a message is encrypted under an NP statement and can only be decrypted with the help of a witness for that statement. In our case, the statement would be that a number of signatures of a known message *exist*. This connection inspired the name of our primitive. However, the security notion differs, as classical witness encryption guarantees security of the ciphertexts only if the statement is *false*.

The security notion we target is conceptually closer to *threshold encryption*, which was first introduced by Desmedt and Frankel [DF90]. In threshold encryption, data is encrypted towards multiple decryption servers, out of which a threshold number needs to collaborate to decrypt. Our scheme can be viewed as an instance of threshold encryption, where collaboration consists of providing signatures.

A major difference is that we require no trusted setup, whereas traditional threshold encryption typically involves a common setup phase in which the servers jointly generate key material. While our SWE constructions rely on specific signature schemes, our notion allows users to *bring their own keys* as long as the keys belong to the supported scheme. Consequently, the set of trusted parties can be selected freely at encryption time.

To go even further, we propose a conceptual shift: We envision relying on signatures that are *already inherently produced* within digital ecosystems to build a threshold-encryption scheme that operates *transparently* in such environments. In systems where it can be *expected* that satisfying certain conditions *will* result in signatures being published by designated parties on specific messages, these parties can be chosen ad hoc as the trusted entities for SWE encryption. This can happen without their active participation in decryption and, more strikingly, *without them even being aware* of SWE encryption happening around them. We will see that this assumption is not only reasonable, but that this approach also opens up additional use cases of threshold encryption.

Applications. We originally proposed the notion of SWE in our paper [DHMW23] to facilitate time-release encryption based on blockchains, which can in turn be applied to settings such as closed-bid auctions or to mitigate front-running. This is facilitated by repurposing the work that committees perform in the maintenance of proof-of-stake (PoS) blockchains. These committees periodically vote on and sign blocks, which appear at roughly regular time intervals. Our aim is to turn these block signatures into time-release decryption tokens. More than just an application of SWE, we consider this a contribution in its own right, and more information on the underlying assumptions and system model will follow after further considerations on SWE.

Another notable application was proposed by Madathil et al. [Mad+23], who introduced a system called *verifiable witness encryption based on threshold signatures* (VweTS), which is similar to SWE, to perform payments conditioned on real-world events. They achieve this by essentially SWE-encrypting transactions to so-called *oracles* on the blockchain, which are third-party providers that attest to real-life events such as weather data or outcomes of sports games by signing them.

Asymptotic Bottlenecks. These envisioned use cases require support for potentially large signer sets due to their decentralized nature. However, the previous SWE/VweTS constructions produce ciphertexts whose size grows linearly with the number of potential signers, which might constitute a bottleneck.

In our follow-up paper [ADMSW24], a joint work with Gennaro Avitabile, Nico Döttling, Bernardo Magri, and Christos Sakkas, we therefore studied the existence of *compact* SWE schemes, where ciphertext size grows sublinearly in the number of potential signers.

Overview of our SWE Constructions. In this thesis, our study of signature-based witness encryption (SWE) proceeds along two complementary directions.

On the one hand, we study the **practicability of SWE** and provide a scheme which is optimized towards concrete efficiency and integration with blockchain systems. This first construction is based on our work in [DHMW23] and described in Chapter 4.

Reflecting our choice to target blockchain-deployment, the SWE scheme is based on BLS as its underlying signature. We make our practical SWE scheme compatible with multi-signatures, albeit for a slightly modified aggregation algorithm than standard BLS multi-signatures. Our security proofs are in the ROM and the size of an SWE encryption grows linearly with the number of potential signers.

We add a layer of non-interactive zero-knowledge (NIZK) proofs to provide verifiability to our practical SWE scheme, which ensures that a given ciphertext consistently opens to a plaintext with claimed properties (e.g., we can make sure that we receive a signed transaction). This opens up more varied use cases on the blockchain.

In the same chapter, we also introduce our McFly protocol, which integrates BLS-based SWE with a proof-of-stake blockchain to achieve time-release encryption. Some more intuition will follow.

On the other hand, we study the **theoretical feasibility of *compact* SWE** in Chapter 5, which is based on our paper [ADMSW24].

We present an SWE construction in the standard model with ciphertexts growing only polylogarithmically in the number of potential signers. Our construction relies on Turing-machine indistinguishability obfuscation and *strongly puncturable signatures*. Compared to regular signature schemes, these allow for an alternate punctured key generation algorithm, such that a specific message cannot have a valid signature under a punctured key.

Time-Release Encryption on the Blockchain: The McFly Protocol. We now discuss our contribution to time-release encryption in a little more detail.

Time-release encryption, as first introduced by Rivest, Shamir, and Wagner in [RSW96], refers to the setting in which a user encrypts a message, not with a decryptor in mind, but such that the message is hidden for a fixed time and may be decrypted only after that time has passed. Traditionally, time-release is achieved either through long non-parallelizable computation, which results in a waste of computational power, or via a trusted third party, which helps in decrypting once the time has passed.

The idea of implementing time-release by using a blockchain with its inherent trust assumptions as a drop-in replacement for a trusted third party is not novel. In fact, already in 2018 Liu, Jager, Kakvi, and Warinschi [LJKW18] proposed using a blockchain as a reference clock for time-release encryption. Their construction employs extractable witness encryption to enable decryption once, relative to the encryption time, a specified number of new blocks have been published in a proof-of-work chain. Further papers suggesting some variation of encryption to the future using a blockchain include [CDKKN22; Ben+20; Gen+21]. However, this previous

line of work relies either on impractical tools or large online communication complexity, rendering it infeasible in practice. Additional details are provided in our related work section § 4.2.3.

What makes our work stand out is that it is the first in this line of work with demonstrable practicability, as we showcase in our paper [DHMW23] that the BLS-based SWE scheme on which McFly hinges is implementable and runs in a reasonable time.

To achieve time-release functionality, we integrate SWE with a *proof of stake* (PoS) blockchain, as described in [DHMW23]. In such blockchains, consensus is typically reached by selecting a subset of users as a committee, which jointly decides which blocks are to be included in the chain. Such a committee is usually selected by a lottery with winning probability proportional to the coins owned by each user on the chain or by eligible committee members applying by locking away large amounts of collateral on the chain, which they are not allowed to spend.

Our observation is that committees *naturally* select and sign blocks as part of maintaining these blockchains. We aim to reuse the trust assumptions and effort that is present in this ecosystem and piggyback time-release encryption on it: messages are SWE-encrypted to the committee keys responsible for signing a future block, such that the corresponding signature(s) become available when that block is published, enabling decryption.

This approach combines the benefits of trusted-party-based and computation-based time-release: Compared to regular time-lock puzzles, where computation needs to be continuously run, here users receiving a ciphertext can simply wait for the correct signature to appear on chain and then decrypt efficiently. However, we will not require any active communication with the parties maintaining the chain. Encryption and decryption using McFly are possible transparently, and users only require read-access to the chain. Otherwise, much like time-lock puzzles, the only communication necessary is sending a single ciphertext.

There are, however, several additional considerations that we need to take into account to make our approach work.

First, at encryption time, we need to already know the committee at decryption time, thus leading to a relatively short horizon of how far into the future we can encrypt.[2] Moreover, SWE requires predictable messages, yet blocks are typically not fully determined in advance. Hence, we suggest that a predictable block header be signed by the committee in addition to the block.

On a technical note, McFly is not compatible with *all* PoS chains. Specifically, McFly relies on byzantine fault tolerance (BFT) blockchains or blockchains with a finality layer (e.g. Ethereum's Casper [BG19] or Afgjort [DMMNT20]), as these guarantee that blocks that consensus has been reached on are *final* in the sense that there can not be a fork of the blockchain later on, replacing this block by a different one. Finality is important to us, as we want time to roughly equate to the number of blocks published, but in case of a fork this becomes less well-defined.

A detailed description of our blockchain model and the McFly protocol can be found in Chapter 4, and in particular in § 4.7.

[2] We notice that concurrent work [GMR23] deployed an almost identical BLS-based scheme in the context of a randomness beacon with a relatively fixed committee using a distributed key-setup. This deployment allows to achieve encryption arbitrarily into the future from the same techniques by using the joint committee key, albeit tolerating a setup phase.

1.4 Additional Work

I want to highlight that I have also worked on other interesting topics during my PhD, which are beyond the scope of this thesis. One such line of work resulted in the paper *Constrained Verifiable Random Functions Without Obfuscation and Friends* [BCGÜW25], which was accepted to TCC 2025 and is joint work with Nicholas Brandt, Miguel Cueto Noval, Christoph U. Günther, and Akin Ünal.

A constrained verifiable random function (CVRF) is an extension of pseudo-random functions (PRFs), which combines two additional properties often useful in distributed systems: While a PRF is a keyed function whose outputs are computationally indistinguishable from random, a CVRF additionally provides *verifiability* and *constrainability*. Verifiability was introduced in the context of VRFs[3] by Micali, Rabin, and Vadhan [MRV99]. It requires the inclusion of a public key that enables public verification of claimed PRF outputs without revealing information about outputs at any other points. Constrainability, introduced in the context of CPRFs [BW13; KPTZ13; BGI14] allows the key generator to partially *delegate* the ability to evaluate the PRF on a constrained subset by generating a constrained key that enables evaluation on the chosen subset without revealing information about function values outside it.

While prefix-constrained PRFs can be directly constructed from the GGM PRF due to Goldreich, Goldwasser, and Micali [GGM84] and VRF constructions exist from well-studied and simple assumptions (e.g., pairing-based VRFs exist due to Lysanskaya [Lys02] or Dodis and Yampolskiy [DY05]), bringing both properties together has long been known to be a delicate task. Although a rich body of work on CVRFs exists [Fuc14; CRV14; LLC15; LZW19; ZLX23; DDM17; Dat20], all prior constructions rely on heavy tools such as multilinear maps, indistinguishability obfuscation, or functional encryption. Our work resolves the long-standing open question of whether CVRFs can be built from simpler assumptions in the plain model.

In [BCGÜW25], we construct a *prefix*-CVRF that can be viewed as a verifiable version of the classic GGM PRF [GGM84]. Our construction relies only on degree-2 algebraic pseudo-random generators (PRGs) and bilinear pairings. Delegation works natively for GGM PRFs by outputting intermediate evaluation values of the tree-based scheme. The key idea to add verifiability is that using a degree-2 polynomial as a PRG allows us to run proofs of correct evaluation of intermediate GGM states "in the exponent", which can be efficiently verified via pairings since only degree-2 polynomials are evaluated.

The main work towards our result lies in its security proof: Within Maurer's Generic Group Model [Mau05], we base security on the Multivariate Quadratic (MQ) problem and a newly introduced variant (MQ+), leveraging new proof techniques.

Regarding potential applications, we note that our construction instantiates the notion of *exponent VRFs* recently proposed by Boneh, Haitner, Lindell, and Segev in [BHLS25] without relying on the ROM. The authors of [BHLS25] propose applications in distributed key generation and hierarchical key derivation for blockchain wallets. Another use case we suggest is off-chain stake-pooling, which may help mitigate centralization in PoS blockchains, which stems from users with little stake in the system rarely winning leader elections and thus being disincentivized from continuously running a computer towards maintenance of the system, leaving committee duty to users who concentrate a large share of on-chain wealth.

[3] In the context of versatility of signatures, we note that *unique* signatures, i.e., signatures that have exactly one valid signature per signer-message pair, already fulfill the somewhat relaxed notion of being a verifiable unpredictable function (VUF) as observed already in [MRV99].

Chapter 2

Preliminaries

2.1 Notation

Throughout this work, we will use some common notation.

$\lambda \in \mathbb{N}$ denotes the security parameter. For $n \in \mathbb{N}$, we use the notation $[n]$ to denote the set $\{1, \ldots, n\}$.

We write $x \leftarrow \mathcal{A}(\mathsf{in}; r)$ to denote that x is the output of running the randomized algorithm $\mathcal{A}$ on input in with randomness $r \leftarrow \{0,1\}^*$. We omit this randomness when it is obvious from context or not required explicitly.

Let S be a (finite) set; then we denote by $x \leftarrow\!\!\$\ S$ that the element x is sampled from the uniform distribution over S. Likewise, if D is any distribution over a set S, then $x \leftarrow\!\!\$\ D$ denotes that the element $x \in S$ is sampled according to D.

Sometimes, when working with vectors, we use boldface as a shorthand, e.g., $\boldsymbol{x} = (x_1, \ldots, x_n)$.

For two algorithms $\mathcal{A}$ and $\mathcal{O}$, $\mathcal{A}^{\mathcal{O}}$ means that $\mathcal{A}$ is run with oracle access to $\mathcal{O}$. That is, during its runtime, it may query $\mathcal{O}$ on any inputs in of its choice, and receives only the outputs $\mathcal{O}(\mathsf{in})$ in a black-box way.

Further, PPT stands for probabilistic polynomial-time and $\mathsf{poly}(x), \mathsf{negl}(x)$ respectively denote any polynomial or negligible function in parameter x. To elaborate further, a function $f(x)$ in x is negligible if it vanishes faster than the inverse of any polynomial in x.

Given two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, we write $\mathcal{D}_1 \approx_c \mathcal{D}_2$ to denote that they are computationally indistinguishable, that is, any PPT distinguisher has negligible probability to tell them apart.

In the following, we will present the cryptographic primitives, models, and assumptions necessary for our main constructions. Note that the following definitions are mostly taken verbatim from [BDW22; DHMW23; ADMSW24].

2.2 Signature Schemes

First, fittingly for a work on signature schemes, let us formally define a standard EUF-CMA secure signature scheme, as informally outlined in the Introduction.

Definition 1 (Signature Scheme)**.** *A signature scheme* Sig *is composed of the following algorithms:*

- $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda; r)$ *takes as input the security parameter* λ *and random coins* $r \in \{0,1\}^\lambda$ *(whenever* r *is omitted, it means it is chosen uniformly at random). It outputs a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sk})$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ *takes as input a signing key* sk *and a message* m*. It outputs a signature* σ.
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, \sigma, m)$ *takes as input a verification key* vk*, a signature* σ *and a message* m*. It outputs a bit* $b \in \{0,1\}$.

A signature scheme needs to have the following properties.

Definition 2 (Correctness)**.** *We say that a signature scheme is correct if for all $\lambda \in \mathbb{N}$ and every message m we have that*

$$\Pr\left[1 \leftarrow \mathsf{Verify}(\mathsf{vk}, \sigma, m) : \begin{array}{c}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{array}\right] = 1.$$

Definition 3 (Existential Unforgeability against Chosen Message Attacks)**.** *We say that a signature scheme is existentially unforgeable against chosen message attacks (EUF-CMA) if for all $\lambda \in \mathbb{N}$ and all adversaries $\mathcal{A}$ we have that*

$$\Pr\left[1 \leftarrow \mathsf{Verify}(\mathsf{vk}, \sigma^*, m^*) : \begin{array}{c}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

where m^* was never queried to the oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

2.3 Hash Functions

Hash functions compress arbitrary-length inputs to fixed-length digests and are designed to behave like "random-looking" maps while remaining efficiently computable and publicly verifiable. They are a cryptographic staple used, for example, in commitment schemes or hash-and-sign signatures. The modern security definition of collision resistance for hash functions dates back to its formalization by Ivan Damgård in 1987 [Dam88].

Definition 4 (Hash Function Family)**.** *A keyed family of hash functions $\mathbb{H}$ consists of the following functions:*

- $k \leftarrow \mathsf{KeyGen}(1^\lambda)$*: The key generation algorithm takes a security parameter and outputs a key k.*
- $h \leftarrow \mathbb{H}_k(m)$*: The hash algorithm takes as input a key k and a message $m \in \{0,1\}^*$. It outputs a hash h.*

We expect a hash function family to be collision resistant.

Definition 5 (Collision Resistance)**.** *A family of hash functions is collision resistant, if for any PPT adversary $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} m_1 \neq m_2 \text{ and } H_k(m_1) = H_k(m_2) : \\ k \leftarrow \mathsf{KeyGen}(1^\lambda); (m_1, m_2) \leftarrow \mathcal{A}(k)\end{array}\right] < \mathsf{negl}(\lambda)$$

In the following, for convenience we will omit the key and assume it has been handed out as a public hash function $H = \mathbb{H}_k$.

2.4 Idealized Models

We will utilize two well-established idealized models in our security analyses and provide an overview here.

2.4.1 Random Oracle Model

We will be using the *random oracle model* (ROM), inroduced by Bellare and Rogaway [BR93], in which a hash function can only be evaluated by directly querying the hashing oracle. The output on a message that has not been queried yet is uniformly

sampled from the hash functions co-domain and determined at the time of the query. This is a heuristic for the behaviour of hash functions that allows us to simulate the hashing oracle ourselves in our reductions.

2.4.2 Common Reference Strings

Another idealized model, which was initially introduced by Blum, Feldman and Micali in [BFM88] in the context of removing interaction in zero-knowledge proofs, is the *common reference string* (CRS) model. In this model, we assume that all parties participating in a protocol have access to a shared, typically short, reference string, which was generated in a trusted setup. If the CRS is sampled as a uniformly random bit string, as in the original work of Blum, Feldman, and Micali [BFM88], we refer to it as an *unstructured* CRS. Alternatively, the CRS may be generated to satisfy certain algebraic properties or to embed hidden trapdoor information, in which case we speak of a *structured* CRS.

2.4.3 Standard Model

We speak of the *plain* or *standard* model, when there is neither a trusted setup nor a random oracle required for our security proofs. Standard model constructions are generally preferable, as they provide more sound guarantees, however not all primitives used today exist in the standard model (e.g. NIZK proofs, see § 2.11.2).

2.5 Proof Technique: Rewinding

Rewinding is a technique used to prove the security of many protocols. In particular, it is often used when analyzing zero-knowledge proof systems.

The main idea is that if we have a simulation interacting with an adversary $\mathcal{A}$ in multiple steps, then if the simulation gets stuck at some point, it is admissible to revert the 'internal state' of the adversary to an earlier point in time and try again.

For example if $\mathcal{A}$ sends a message a_1, the simulator responds with a challenge c_1 which is followed by a message a_2 from $\mathcal{A}$, then if we do not like this response or need to know an additional response, we can 'go back in time' and rerun the same execution of $\mathcal{A}$ from the point in time when it had output a_1 and we simply input a different challenge c_1' to receive a different output a_2'.

This is useful in conjunction with the random oracle model, where a challenge might consist of the value output by the random oracle on the previous inputs. We might want to rewind and reprogram the random oracle to receive a different output behaviour in our reductions.

2.5.1 Expected-Time Forking Lemma

To model the success probability and runtime of a rewinding procedure, we will use the elegant and powerful forking lemma that was proven secure by Bellare and Neven in [BN06]:

Lemma 1 (Expected-Time Forking Lemma)**.** *Let H be a set of size $h \geq 2$, $q \geq 1$ an integer and $\mathcal{A}$ a randomized algorithm, that on input $x, h_1, h_2, \ldots, h_q$ returns an integer from $0, \ldots, q$ in at most $T_{\mathcal{A}}(|x|)$ time steps. Let $\mathcal{B}$ be a brute-force algorithm, that takes input x and halts after at most $T_{\mathcal{B}}(|x|)$ steps, outputting 1.*

Let the accepting probability $P_{\mathcal{A}}(x)$ be defined as the probability of the following experiment outputting 1:

Randomly choose coins ρ for $\mathcal{A}$

Pick $h_1, \ldots, h_q \leftarrow_\$ H$ *and set* $i \leftarrow \mathcal{A}(x, h_1, \ldots, h_q; \rho)$
If $i \geq 1$*, return* 1*, else return* 0*.*

Then it holds that the forking algorithm $F_{\mathcal{A}}(x)$ *given as:*

Randomly choose coins ρ *for* $\mathcal{A}$
Pick $h_1, \ldots, h_q \leftarrow_\$ H$ *and set* $i \leftarrow \mathcal{A}(x, h_1, \ldots, h_q; \rho)$
If $i = 0$*, return 0.*
Repeat, in parallel with $\mathcal{B}(x)$*:*
  Pick $h'_i, \ldots, h'_q \leftarrow_\$ H$
  Set $j \leftarrow \mathcal{A}(x, h_1, \ldots, h_{i-1}, h'_i, \ldots, h'_q; \rho)$
Until $(i = j$ *and* $h'_i \neq h_i)$ *or* $\mathcal{B}$ *has halted.*
Return 1

returns 1 *with the same probability* $P_{\mathcal{A}}(x)$ *in expected time* $T_{F_{\mathcal{A}}}(|x|) \leq (4q+1)T_{\mathcal{A}}(|x|) + \frac{4q}{h}T_{\mathcal{B}}(|x|)$*.*

2.6 Bilinear Group Setting and Assumptions

Bilinear pairings were introduced as a new assumption to cryptography by Joux in [Jou00] and enabled a multitude of constructions. Galbraith, Paterson, and Smart [GPS08] put forth a good reference framework for the blackbox use of bilinear pairings. Following their nomenclature, we use Type-3 pairings, which are commonly called *asymmetric* bilinear groups. Since we only use this type of bilinear groups, we will often simply refer to them as bilinear groups.

Definition 6 (Asymmetric Bilinear Groups)**.** *An asymmetric* bilinear group*, is a triple of cyclic groups* $\mathbb{G}_1$*,* $\mathbb{G}_2$*,* $\mathbb{G}_T$ *of prime order* p *with their respective generators* g_1*,* g_2 *and* g_T *and a map* $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *s.t. the following holds:*

- *The map* e *is bilinear, i.e.,* $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ *for all* $a, b \in \mathbb{Z}$
- *It is non-degenerate, i.e.,* $e(g_1, g_2) \neq 1$*.*

We further assume that the group operations in all of these groups as well as e *can be computed in one time step.*

Parameter Settings. We usually work with a security parameter λ to parametrize security. In terms of our groups, we will commonly assume that they have a common prime order p, where p is a λ-bit prime. In particular, this allows us to treat $1/p$ as negligible in λ throughout this work.

2.6.1 Assumptions

We will generally assume asymmetric bilinear groups of Type-3 to fulfill both the computational Co-Diffie-Hellman and the decisional bilinear Diffie-Hellman assumption as defined below, but we will explicitly state in our theorem statements, which assumptions are required for individual results.

Definition 7 (Computational Co-Diffie-Hellman, (compare [CHKM10]: co-DHP*))**.** *The computational Co-Diffie-Hellman (co-CDH) assumption for a pair of groups* $(\mathbb{G}_1, \mathbb{G}_2)$ *states that the probability*

$$Pr\left[\mathcal{A}(g_1, g_1^x, g_2, g_2^x, h_1) = h_1^x : x \leftarrow_\$ \mathbb{Z}_p, h_1 \leftarrow_\$ \mathbb{G}_1\right]$$

is negligible for polynomial adversaries $\mathcal{A}$*, where* $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ *are generators.*

Definition 8 (Decisional Bilinear Diffie-Hellman (compare [CM11]: DBDH-3))**.** *The bilinear Diffie-Hellman (BDH) assumption for a triple of groups* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *of order* p *states that the following distributions are computationally close:*

$$(g_1, g_1^x, g_1^\alpha, g_2, g_2^x, g_2^r, g_T^{\alpha x r}) \approx_c (g_1, g_1^x, g_1^\alpha, g_2, g_2^x, g_2^r, g_T^y),$$

where $x, y, \alpha, r \leftarrow_\$ \mathbb{Z}_p$ *and* $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, g_T = e(g_1, g_2) \in \mathbb{G}_T$ *are generators.* $e : \mathbb{G}_1 \bigtimes \mathbb{G}_2 \to \mathbb{G}_T$ *is a bilinear map.*

2.7 Symmetric Encryption Schemes

Symmetric Encryption is an essential building block of cryptography: Two parties with a shared key can securely send messages in the presence of an eavesdropper. We consider schemes achieving IND-CPA security, a standard security notion for confidentiality under chosen-plaintext attacks; see, for example, chapter 3 in the well-known *Introduction to Modern Cryptography* by Katz and Lindell [KL14] for formal definitions and discussion.

Definition 9 (Symmetric Encryption)**.** *A symmetric encryption scheme* SKE *is a tuple of three efficient algorithms* $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ *such that*

- $K \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$*: The probabilistic key generation algorithm on input the security parameter* 1^λ *outputs a key* $K \in \{0,1\}^\lambda$*.*
- $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(K, m)$*: The probabilistic encryption algorithm takes as input the key* K *and a message* $m \in \{0,1\}^*$ *and outputs a ciphertext* $\mathsf{ct} \in \{0,1\}^*$*.*
- $m \leftarrow \mathsf{SKE.Dec}(K, \mathsf{ct})$*: The deterministic decryption algorithm takes as input the key* K *and a ciphertext* ct *and outputs a message* m*.*

We require a symmetric encryption scheme to fulfill correctness, IND-CPA security and pseudorandom ciphertexts as defined below:

Definition 10 (Correctness)**.** *We say that a symmetric encryption scheme is correct, if for all* $\lambda \in \mathbb{N}$ *and all* $m \in \{0,1\}^*$*, we have*

$$\Pr\left[\mathsf{SKE.Dec}(K, \mathsf{SKE.Enc}(K, m)) = m \mid K \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)\right] = 1.$$

Definition 11 (Security)**.** *We say that a symmetric encryption scheme is IND-CPA secure, if no adversary* $\mathcal{A}$ *has more than negligible advantage in the following experiment* $\mathsf{Exp}_{\mathsf{IND\text{-}CPA}}(\mathcal{A}, 1^\lambda)$*:*

- *The experiment samples* $b \leftarrow_\$ \{0,1\}$ *and* $K \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$*.*
- $\mathcal{A}$ *gets access to an encryption oracle* $O(K, \cdot)$*, which on input* m *outputs* $\mathsf{Enc}(K, m)$*, which it may use during the whole experiment.*
- $\mathcal{A}$ *chooses two challenge messages* m_0, m_1 *with* $|m_0| = |m_1|$*.*
- *The challenger sends* $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(K, m_b)$ *to* $\mathcal{A}$*.*
- $\mathcal{A}$ *outputs a bit* b'*.*

The advantage of $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{IND\text{-}CPA}} := \left|\Pr\left[\mathsf{Exp}_{\mathsf{IND\text{-}CPA}}(\mathcal{A}, 1^\lambda)\right] - \frac{1}{2}\right|.$$

2.8 Pseudorandomness

2.8.1 Pseudorandom Generators

We recall the definition of pseudorandom generators (PRG). They take a short truly random key sometimes also called seed and generate a polynomial number of pseudorandom bits, that is bits that are indistinguishable from truly random bits. They can be instantiated based on one-way functions [Lev85; HILL99].

Definition 12 (Pseudorandom Generators)**.** *Let $\alpha = \alpha(\lambda)$ and $\beta = \beta(\lambda)$ be two polynomials. A pseudorandom generator $\mathsf{PRG} : \{0,1\}^\alpha \to \{0,1\}^\beta$ is a function such that for all PPT adversaries $\mathcal{A}$ we have that*

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\nu) : \begin{array}{c} s \leftarrow\!\!\$\ \{0,1\}^\alpha \\ \nu \leftarrow \mathsf{PRG}(s) \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}(\nu) : \nu \leftarrow\!\!\$\ \{0,1\}^\beta\right]\right| \leq \mathsf{negl}(\lambda).$$

2.8.2 Pseudorandom Functions

Pseudorandom functions (PRF) were first introduced by Goldreich, Goldwasser and Micali in [GGM84], where they show how to construct them from PRGs. The idea of a PRF is that it behaves like a random function given a short truly secret key – in contrast to PRGs which stretch a random key to a polynomial amount of pseudorandom bits, such a function can be queried on exponentially many inputs, thus encoding a larger space of pseudorandom bits out of which a PPT adversary may query polynomially many in a random access manner.

Definition 13 (Pseudorandom Functions)**.** *Let $\alpha = \alpha(\lambda)$ and $\beta = \beta(\lambda)$ be two polynomials. A keyed family of functions $\mathsf{PRF}_K : \{0,1\}^\alpha \to \{0,1\}^\beta$ for keys $K \in \{0,1\}^\lambda$ is a pseudo-random function (PRF) family, if*

- *given K, m the function $\mathsf{PRF}_K(m)$ is efficiently computable and*
- *for every PPT distinguisher $\mathcal{D}$, it holds $\mathcal{D}^{\mathsf{PRF}_K(.)} \approx_c \mathcal{D}^{F(.)}$, where $K \leftarrow\!\!\$\ \{0,1\}^\lambda$ and F is chosen randomly from all functions from $\{0,1\}^{\alpha(\lambda)}$ to $\{0,1\}^{\beta(\lambda)}$.*

We also write $\mathsf{PRF}(K, m)$ for $\mathsf{PRF}_K(m)$.

2.8.3 Puncturable Pseudorandom Functions

A useful variation of pseudorandom functions are *puncturable pseudorandom functions* (PPRFs). They add a puncturing algorithm which takes a PRF key K and a set S of up to polynomially many points in its domain and outputs a punctured key K_S which allows one to evaluate the PRF on any input $x \notin S$. This notion has been demonstrated by Sahai and Waters in [SW14] to be an excellent tool to be used together with indistinguishability obfuscation, a notion formalized by Barak et al. [Bar+01]. For more information on indistinguishability obfuscation, we refer the reader to § 2.14.

Definition 14 (Puncturable PRF)**.** *Let $\alpha = \alpha(\lambda)$ and $\beta = \beta(\lambda)$ be two polynomials. A puncturable PRF (PPRF) scheme $\mathsf{PPRF}_{\alpha,\beta} = \mathsf{PPRF}$ is composed of the following algorithms:*

- *$K \leftarrow \mathsf{KeyGen}(1^\lambda)$ takes as input a security parameter λ. It outputs a key K.*
- *$y \leftarrow \mathsf{Eval}(K, x)$ takes as input a key K and $x \in \{0,1\}^\alpha$. It outputs $y \in \{0,1\}^\beta$.*
- *$K_S \leftarrow \mathsf{Punct}(K, S)$ takes as input a key K and a subset $S \subseteq \{0,1\}^\alpha$. It outputs a punctured key K_S.*
- *$y \leftarrow \mathsf{EvalPunct}(K_S, x)$ takes as input a punctured key K_S and $x \in \{0,1\}^\alpha$. It outputs $y \in \{0,1\}^\beta$.*

We expect such a scheme to be pseudorandom *and* correct.

Definition 15 (Correctness)**.** *A PPRF scheme* PPRF *is said to be correct if it preserves functionality under puncturing. That is, for all* $\lambda \in \mathbb{N}$*, for all* $S \subseteq \{0,1\}^{\alpha}$*, all* $x \notin S$ *we have that*

$$\Pr\left[\mathsf{Eval}(K,x) = \mathsf{EvalPunct}(K_S,x) : \begin{array}{c} K \leftarrow \mathsf{KeyGen}(1^\lambda) \\ K_S \leftarrow \mathsf{Punct}(K,S) \end{array}\right] = 1.$$

Definition 16 (Pseudorandomness)**.** *A PPRF scheme* PPRF *is said to be pseudorandom if for all* $\lambda \in \mathbb{N}$ *we have that:*
1) PPRF.Eval *is a PRF with respect to keys drawn from* PPRF.KeyGen*, i.e., for every PPT distinguisher* $\mathcal{D}$*, it holds* $\mathcal{D}^{\mathsf{Eval}(K,.)} \approx_c \mathcal{D}^{F(.)}$*, where* $K \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* F *is chosen randomly from all functions from* $\{0,1\}^{\mu(\lambda)}$ *to* $\{0,1\}^{\nu(\lambda)}$*.*
2) PPRF *is pseudorandom at punctured points. That is, it holds for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}_2(K_S,S,T,\mathsf{aux}) : \begin{array}{c} (S,\mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda);\ K \leftarrow \mathsf{KeyGen}(1^\lambda) \\ K_S \leftarrow \mathsf{Punct}(K,S);\ T \leftarrow \mathsf{Eval}(K,S) \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}_2(K_S,S,T,\mathsf{aux}) : \begin{array}{c} (S,\mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda);\ K \leftarrow \mathsf{KeyGen}(1^\lambda) \\ K_S \leftarrow \mathsf{Punct}(K,S);\ T \leftarrow\!\!\$\ \{0,1\}^{\beta|S|} \end{array}\right] \right| \leq \mathsf{negl}(\lambda).$$

Puncturable PRFs that can be punctured at $|S| = \mathsf{poly}(\lambda)$ points can be constructed from one-way functions. The size of the punctured key is $\mathcal{O}(|S| \cdot \mathsf{poly}(\lambda))$.

This is argued to hold, e.g., by Boyle, Goldwasser, and Ivan in [BGI14] where it is observed that such a construction can be made by altering the GGM PRF [GGM84].

2.9 Linear Secret Sharing

Linear secret sharing (LSS) allows a dealer to divide a secret among multiple parties such that only authorized subsets can reconstruct the secret, while unauthorized subsets learn nothing. In this work, we use threshold LSS, where any set of at least t out of n parties can jointly reconstruct the secret. The archetypal example of an LSS is Shamir's secret sharing [Sha79], which is not only a foundational construction, but further, as observed by McEliece and Sarwate [MS81], illustrates the connection between secret sharing and error-correcting codes, specifically its shares form codewords of Reed-Solomon codes.

We will first give an abstract definition of a linear secret sharing scheme, followed by a detailed treatment of Shamir's original scheme.

Definition 17 (Linear Secret Sharing)**.** *Let* $t \leq n$*. A* (t,n)*-linear secret sharing (LSS)* LSS *scheme is composed of the following algorithms:*

- $(s_1,\ldots,s_n) \leftarrow \mathsf{Share}(m)$ *takes as input a message* m*. It outputs* n *shares* $(s_1,\ldots,s_n)$*.*
- $m \leftarrow \mathsf{Reconstruct}(s_{i_1},\ldots,s_{i_t})$ *takes as input* t *shares* $(s_{i_1},\ldots,s_{i_t})$*. It outputs a message* m*.*
- $(s_{i_{z+1}},\ldots,s_{i_n}) \leftarrow \mathsf{RemainShare}(m,s_{i_1},\ldots,s_{i_z})$ *that takes as input a message* m *and uniformly chosen shares* $s_{i_j} \leftarrow\!\!\$\ \{0,1\}^\lambda$ *for* $j \in [z]$ *with* $z < t$*, and outputs the remaining shares* $(s_{i_{z+1}},\ldots,s_{i_n})$*.*

Definition 18 (Correctness)**.** *A LSS scheme* LSS *is said to be correct if for all messages* m *and all subsets* $\{i_1,\ldots,i_t\} \subseteq [n]$

$$\Pr\left[m = \mathsf{Reconstruct}(s_{i_1},\ldots,s_{i_t}) : (s_1,\ldots,s_n) \leftarrow \mathsf{Share}(m)\right] = 1.$$

Moreover, for $z < t$, the following probability is 1:

$$\Pr\left[m = \mathsf{Reconstruct}(s_{i_1}, \ldots, s_{i_t}) : \begin{array}{c} s_{i_j} \leftarrow\!\!\$\ \{0,1\}^\lambda \text{ for } j \in [z] \\ (s_{i_{z+1}}, \ldots, s_{i_n}) \leftarrow \mathsf{RemainShare}(m, s_{i_1}, \ldots, s_{i_z}) \end{array}\right].$$

Definition 19 (Privacy)**.** *We say that a (t, n)-LSS scheme* LSS *is t-private if for all subsets $\{i_1, \ldots, i_z\} \subset [n]$ where $z < t$, all pairs of messages (m_0, m_1) and all PPT adversaries $\mathcal{A}$ we have that*

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(s_{0,i_1}, \ldots, s_{0,i_z}) : (s_{0,1}, \ldots, s_{0,n}) \leftarrow \mathsf{Share}(m_0)\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(s_{1,i_1}, \ldots, s_{1,i_z}) : (s_{1,1}, \ldots, s_{1,n}) \leftarrow \mathsf{Share}(m_1)\right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

Remark. The existence of a RemainShare algorithm that takes less than t shares and a target message m and outputs additional shares to complete the input shares to a sharing of m is not typically required in the definition of a linear secret sharing scheme and we will sometimes omit it, when not required for a specific construction. However we will see that Shamir's secret sharing already naturally supports such a functionality, illustrating feasibility.

2.9.1 Shamir's Secret Sharing

Let $\mathbb{Z}_p$ be the finite field of prime order p and fix distinct elements $\xi = \xi_1, \ldots, \xi_n \in \mathbb{Z}_p$.

$\mathsf{Shamir.Share}(m)$ picks a random degree $t-1$ polynomial f with $f(0) = m$ and sets $s_i = f(\xi_i)$ for $i \in [n]$ as its shares.

To reconstruct, we use *Lagrange Interpolation*: For a set of supporting points $\xi_1, \ldots, \xi_k$ from a finite field $\mathbb{Z}_p$, where $p \in \mathbb{N}$ is prime, the Lagrange basis polynomials are given by $L_1, \ldots, L_k$, where

$$L_i(x) = \prod_{j \in [k]; j \neq i} \frac{x - \xi_j}{\xi_i - \xi_j}.$$

These are chosen such that $L_i(\xi_j) = 1$ iff $i = j$ and 0 otherwise. Consequently, given a set of k data points (ξ_i, y_i), we can output a polynomial $f_L(x) = \Sigma_{i \in [k]} L_i(x) \cdot y_i$ that will run through these points and which has degree at most k-1. This process is called Lagrange Interpolation. Hence, if $k > t-1$, we will get back the original polynomial used in sharing and can evaluate $f(0) = m$.

2.9.2 Relationship with Reed-Solomon Codes

We will introduce Reed-Solomon Codes [RS60] and their relationship with Shamir's secret sharing very briefly; a more in depth explanation can be found in [Hal10].

Let $\mathbb{Z}_p$ be the finite field of prime order p and parameters n, k with $1 \leq k \leq n < p$. Fix distinct elements $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n) \in \mathbb{Z}_p^n$. The Reed-Solomon code $RS_{n,k}[\boldsymbol{\xi}]$ encodes messages of length k into codewords of length n and consists of all vectors $\boldsymbol{c} = (f(\xi_1), \ldots, f(\xi_n))$ for any polynomial $f(X) = \sum_{i=0}^{k-1} a_i X^i$ of degree $k-1$.

This code is generated by the Vandermonde matrix $\mathbf{G} \in \mathbb{Z}_p^{n \times k}$

$$\mathbf{G} = \begin{pmatrix} 1 & \xi_1 & \xi_1^2 & \ldots & \xi_1^{k-1} \\ 1 & \xi_2 & \xi_2^2 & \ldots & \xi_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_n & \xi_n^2 & \ldots & \xi_n^{k-1} \end{pmatrix}$$

and has a parity-check matrix $\mathbf{H} \in \mathbb{Z}_p^{(n-k)\times n}$ with the property that a vector $\boldsymbol{c} \in \mathbb{Z}_p^n$ is a codeword of $RS_{n,k}[\boldsymbol{\xi}]$ if and only if $\mathbf{H}\boldsymbol{c} = 0$

$$\mathbf{H} = \begin{pmatrix} v_1 & v_2 & \dots & v_n \\ v_1 \cdot \xi_1 & v_2 \cdot \xi_2 & \dots & v_n \cdot \xi_n \\ v_1 \cdot \xi_1^2 & v_2 \cdot \xi_2^2 & \dots & v_n \cdot \xi_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ v_1 \cdot \xi_1^{n-k-1} & v_2 \cdot \xi_2^{n-k-1} & \dots & v_n \cdot \xi_n^{n-k-1} \end{pmatrix} \quad \text{where} \quad v_i = \frac{1}{\prod_{j \neq i}(\xi_j - \xi_i)}.$$

We see, that a set of shares from Shamir's secret sharing scheme for k out of n shares can be computed as $(s_1, \dots, s_n) = \mathbf{G}(m, r_2, \dots, r_k)^\top$ where the r_i are randomly chosen. This represents evaluating the random polynomial $f(x) = m + \sum_{i=2}^{k} r_i x^{i-1}$ on the points ξ_i. Hence, any correctly generated set of shares lies in $RS_{n,k}[\boldsymbol{\xi}]$. This relationship was pointed out in [MS81].

The fact that $\mathbf{H}$ is a parity-check matrix for this code implies that we can check whether a set of shares $(s_1, \dots, s_n)$ forms a correct Shamir sharing by testing whether $\mathbf{H}(s_1, \dots, s_n)^\top = 0$.

This description of Shamir's sharing also lets us easily see that we can compute $\mathsf{RemainShare}$ as outlined before. If we are given an incomplete set of shares $(\bar{s_{i_1}}, \dots, \bar{s_{i_z}})$ $z < k$ and a desired encoded message m, then we know we must find $r_2, \dots, r_k$ and $s_1, \dots, s_n$ such that

$$\mathbf{G} \begin{pmatrix} m \\ r_2 \\ \vdots \\ r_k \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix}.$$

Additionally we require $s_{i_j} = \bar{s_{i_j}}$ for the given shares.

Given that all ξ_i are distinct, the Vandermonde matrix has rank k. We impose at most k linear constraints by fixing z shares and the message m, thus we can solve this system for the remaining shares.

2.10 Commitment Schemes

Commitment schemes are a versatile tool in cryptography. They allow a user to publicly commit to a value, such that this value is hidden, but the user may later reveal that value and provide a convincing proof, that it was indeed the committed value. This is for example useful in online bidding, where the bids should remain secret until the bidding phase is over.

We will regard commitment schemes in two settings: With a setup producing a common reference string and without setup.

2.10.1 Commitment Schemes in the CRS Model

Definition 20 (Commitment Schemes)**.** *A commitment scheme* $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Vrfy})$ *is composed of the following algorithms:*

- $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes the security parameter* λ *and outputs a common reference string* CRS*.*
- $(\mathsf{com}, \gamma) \leftarrow \mathsf{Commit}(\mathsf{CRS}, m)$*: The commit algorithm takes as input a common reference string* CRS *and a message* m*. It outputs a commitment* com *and opening information* γ*.*

- $b \leftarrow \mathsf{Vrfy}(\mathsf{CRS}, \mathsf{com}, m, \gamma)$: *The verification algorithm takes as input a common reference string* CRS*, a commitment* com*, a message* m *and opening information* γ*. It outputs a bit* $b \in \{0,1\}$*.*

It is expected to fulfill correctness and (computational or perfect) hiding and (computational or perfect) binding.

Definition 21 (Correctness)**.** *We say that a commitment scheme is correct if for all* $\lambda \in \mathbb{N}$*,* $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$ *and every message* m *we have that*

$$\Pr\left[1 \leftarrow \mathsf{Vrfy}(\mathsf{CRS}, \mathsf{com}, m, \gamma) : (\mathsf{com}, \gamma) \leftarrow \mathsf{Commit}(\mathsf{CRS}, m)\right] = 1.$$

Definition 22 (Computational Hiding)**.** *We say that a commitment scheme is computationally hiding if for all* $\lambda \in \mathbb{N}$*,* $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$ *and all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *we have that :*

$$\left| \Pr\left[b = b' \ : \begin{array}{c} (m_0, m_1, aux) \leftarrow \mathcal{A}_1(\mathsf{CRS}) \\ b \leftarrow\!\!\$\ \{0,1\} \\ (\mathsf{com}, \gamma) \leftarrow \mathsf{Commit}(\mathsf{CRS}, m_b) \\ b' \leftarrow \mathcal{A}_2(\mathsf{com}, aux) \end{array} \right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

Definition 23 (Perfect Hiding)**.** *We say that a commitment scheme is perfectly hiding if the probability in Def. 22 is zero for any unbounded adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*.*

Definition 24 (Computational Binding)**.** *We say that a commitment scheme is computationally binding if for all* $\lambda \in \mathbb{N}$*,* $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$ *and all PPT adversaries* $\mathcal{A}$ *we have that*

$$\Pr\left[m_0 \neq m_1 \wedge b = b' = 1 : \begin{array}{c} (\mathsf{com}, m_0, \gamma_0, m_1, \gamma_1) \leftarrow \mathcal{A}(\mathsf{CRS}) \\ b \leftarrow \mathsf{Vrfy}(\mathsf{CRS}, \mathsf{com}, m_0, \gamma_0) \\ b' \leftarrow \mathsf{Vrfy}(\mathsf{CRS}, \mathsf{com}, m_1, \gamma_1) \end{array} \right] = \mathsf{negl}(\lambda).$$

Definition 25 (Perfect Binding)**.** *We say that a commitment scheme is perfectly binding if the probability in Def. 24 is zero for any unbounded adversary* $\mathcal{A}$*.*

2.10.2 Non-Interactive Commitment Schemes

Sometimes we wish to use commitments in a setting that does not allow us to use a common reference string. In that case, we want to skip the Setup phase and instead directly create a commitment for a given message. A commitment scheme allowing to do this is called *non-interactive*.

Definition 26 (Non-Interactive Commitment Schemes)**.** *A non-interactive commitment scheme is a tuple* $\mathsf{CS} = (\mathsf{Commit}, \mathsf{Vrfy})$ *with*

- $(\mathsf{com}, \gamma) \leftarrow \mathsf{Commit}(1^\lambda, m)$*: The commit algorithm takes as input the security parameter* λ *and a message* m*. It outputs a commitment* com *and opening information* γ*.*
- $b \leftarrow \mathsf{Vrfy}(\mathsf{com}, m, \gamma)$*: The verification algorithm takes as input a commitment* com*, a message* m *and opening information* γ*. It outputs a bit* $b \in \{0,1\}$*.*

for which the tuple $(\mathsf{Setup}(1^\lambda) = \mathsf{id}(1^\lambda), \mathsf{Commit}, \mathsf{Vrfy}'(\mathsf{CRS}, \mathsf{com}, m, \gamma))$ *with verification defined as* $\mathsf{Vrfy}'(\mathsf{CRS}, \mathsf{com}, m, \gamma) = \mathsf{Vrfy}(\mathsf{com}, m, \gamma)$ *forms a commitment scheme with CRS in the sense of Def. 20. Here,* id *denotes the identity function, which outputs exactly its input.*

2.10.3 Noteworthy Commitment Schemes

2.10.3.1 Non-Interactive Scheme Based on Goldreich-Levin Hardcore Bits

It is folklore knowledge that there exists a perfectly binding, computationally hiding non-interactive commitment scheme assuming only injective one-way functions (mentioned e.g. in [LS19; BDHKS19]). In [LS19] non-interactive key agreements are further discussed, and it is shown that they follow also from any secure key agreement protocol.

To the best of my knowledge, the idea for this folklore result dates back to work due to Blum [Blu81], where the notion of a *completely secure one-way function* was informally described. Today we would consider that to be a one-way function[1] $f(x)$ which has a one-bit hardcore predicate $B(x)$. Such a function should informally be one-way (i.e., given x, the value $f(x)$ is efficiently computable, but given $f(x)$ the pre-image x should be hard to compute) and additionally, given only $f(x)$ a PPT adversary should only be able to determine $B(x)$ with advantage at most negligibly higher than guessing a bit randomly.

In [Blu81] it is argued that an injective completely secure one-way function can be used to securely compute shared randomness between two parties in the sense that neither party can undetectedly skew the bias of the outcome bit. To choose one bit, the first party chooses some input x and sends $f(x)$ to the second party, who must guess $B(x)$. The first party will after receiving this guess reveal x and whether the guess was correct will determine the joint random bit.

This implicitly makes use of the fact that $f(x)$ binds the first party to the input x, due to the injectivity and the fact that it is hiding x (and $B(x)$) from the second party due to the completely secure one-wayness, thus making the connection to what we consider as a commitment scheme.

In [GL89] Goldreich and Levin showed that any one-way function can be modified to a one-way function which has a hard-core one-bit predicate. Specifically, if $f(x)$ is one-way, then so is $f'(x,r) = f(x)||r$ (with $|x| = |r|$) and $B(x,r) = \langle x, r\rangle_2 = \sum_i (x_i r_i)$ is shown to be a hard-core bit of f'.

Thus, we get a non-interactive commitment scheme from assuming only one-way functions.

2.10.3.2 Pederson Commitments

A concrete commitment scheme that is geared more towards practical efficiency is the Pederson commitment scheme [Ped92], which uses a CRS and is perfectly hiding and computationally binding under the discrete logarithm assumption.

Definition 27 (Pedersen Commitment)**.** *As public parameter we require a group $\mathbb{G}$ of prime order p in which computing discrete logarithms is hard.*

- $\mathsf{Setup}(1^\lambda)$*: Output* $(g,h) \leftarrow_\$ \mathbb{G}^2$
- $\mathsf{Commit}(\mathsf{CRS} = (g,h), m)$*:*
  - *Randomly draw* $r \leftarrow_\$ \mathbb{Z}_p$
  - *Output commitment* $\mathsf{com} = h^r g^m$ *and opening* $\gamma = r$
- $\mathsf{Vrfy}(\mathsf{CRS} = (g,h), \mathsf{com}, m, \gamma)$*: Check whether* $\mathsf{com} = h^\gamma g^m$ *holds. If so output* 1*, else* 0*.*

There are two properties of this scheme which are interesting for our work:

[1] The first formalization of one-way functions was given by Yao in [Yao82]. One way functions are widely considered to be a minimum viable assumption to build meaningful cryptography.

1. The Pedersen Commitment is homomorphic in the sense that for scalars $\alpha, \beta \in \mathbb{Z}_p$, $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$, messages $m_1, m_2 \in \mathbb{Z}_p$ and randomness $r_1, r_2 \in \mathbb{Z}_p$ it holds $\mathsf{Commit}(\mathsf{CRS}, m_1; r_1)^\alpha \cdot \mathsf{Commit}(\mathsf{CRS}, m_2; r_2)^\beta = \mathsf{Commit}(\mathsf{CRS}, \alpha m_1 + \beta m_2, \alpha r_1 + \beta r_2)$
2. Pedersen Commitments are highly compatible with an efficient zero-knowledge proof system called Bulletproofs. Specifically, Bulletproofs enable us to prove NP statements over the committed value of a Pedersen commitment. For more information see our introduction on Bulletproofs in § 2.11.5.2.

2.11 Proof Systems

We will require different sorts of proof systems for our constructions.

We regard a set $\mathcal{X}$ of potential statements and $\mathcal{L} \subseteq \mathcal{X}$ an NP language with witness space $\mathcal{W}$ and witness relation $\mathcal{R}$, i.e., $\mathcal{L} = \{x \in \mathcal{X} : \exists w \in \mathcal{W} \text{ s.t. } \mathcal{R}(x, w) = 1\}$.

Now to show that a statement x is part of the language $\mathcal{L}$ it suffices to provide a verifying witness w. However, when proving cryptographic statements, the witness might often be privacy-sensitive.

The common point of the cryptographic proof systems we will discuss, is to show that a statement x is true, while providing some level of privacy for the witness w.

We only regard non-interactive proofs, that is a prover wishing to show a statement x is in a language $\mathcal{L}$ merely sends one message with a proof π to the receiver, which that person may use to verify membership of x in $\mathcal{L}$ with no further communication required.

2.11.1 Non-Interactive Witness Indistinguishable Proofs

The notion of witness-indistinguishability means that if there are different witnesses w_1, w_2 for the same statement x, then the receiver of a proof should be clueless as to which witness was used in making a proof for x.

Definition 28 (Non-Interactive Witness-Indistinguishable Proofs)**.** *Let $\mathcal{L}$ be an* NP *language. A non-interactive witness-indistinguishable (NIWI) proof system* NIWI *for language $\mathcal{L}$ is composed of the following algorithms:*

- $\pi \leftarrow \mathsf{Prove}(1^\lambda, x, w)$ *takes as input a security parameter λ, a statement $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$. It outputs a proof π.*
- $b \leftarrow \mathsf{Verify}(x, \pi)$ *takes as input a statement $x \in \mathcal{X}$ and a proof π. It outputs a bit $b \in \{0, 1\}$.*

We expect such a NIWI scheme to be perfectly correct*,* perfectly sound *and* witness-indistinguishable*.*

Definition 29 (Perfect Completeness)**.** *We say that a NIWI proof system is perfectly complete if for all $\lambda \in \mathbb{N}$, all statements $x \in \mathcal{X}$ and all witnesses $w \in \mathcal{W}$, if $\mathcal{R}(x, w) = 1$, then*

$$\Pr\left[1 \leftarrow \mathsf{Verify}(x, \pi) : \pi \leftarrow \mathsf{Prove}(1^\lambda, x, w)\right] = 1.$$

Definition 30 (Perfect Soundness)**.** *We say that a NIWI proof system has perfect soundness if for all $\lambda \in \mathbb{N}$, all statements $x \notin \mathcal{L}$ and all proofs π we have that*

$$\Pr\left[0 \leftarrow \mathsf{Verify}(x, \pi)\right] = 1.$$

Definition 31 (Witness-Indistinguishability)**.** *We say that a NIWI proof system is witness-indistinguishable if for all $\lambda \in \mathbb{N}$ and all adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have*

that

$$\left|\Pr\left[b = b' : \begin{array}{l}(x, w_0, w_1, aux) \leftarrow \mathcal{A}_1(1^\lambda);\ b \leftarrow\!\!\$\ \{0,1\} \\ \pi \leftarrow \mathsf{Prove}(1^\lambda, x, w_b);\ b' \leftarrow \mathcal{A}_2(\pi, aux)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

NIWI schemes can be constructed from pairing assumptions [GOS06a], i$\mathcal{O}$ [BP15] or derandomization assumptions [BOV03].

Proof size. Let $\mathcal{C}_x$ be the verification circuit for statement x and a certain relation $\mathcal{R}$, that is, $1 \leftarrow \mathcal{C}_x(w)$ if and only if $\mathcal{R}(x, w) = 1$. Known NIWI schemes for a relation $\mathcal{R}$ achieve a proof size $|\pi| \in \mathcal{O}\left(|\mathcal{C}_x| \cdot \mathsf{poly}(\lambda)\right)$ where $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$.

2.11.2 Non-Interactive Zero-Knowledge Proofs in the Random Oracle Model

There is a stronger notion of privacy called zero-knowledge which demands that the receiver of the proof learns essentially nothing about the witness used – this is formally modelled in terms of a simulation game.

It is commonly accepted that NIZK proof systems for general NP languages cannot exist in the plain model[2] and thus NIZK proofs are constructed in the ROM or common reference string (CRS) model. Blum, Feldman and Micali pioneered the study of NIZK proofs for a single statement in the presence of a common reference string (CRS) [BFM88] and later Blum, De Santis, Micali, and Persiano gave the first NIZK proof system in the CRS model for multiple statements [BDSMP91]. Both constructions follow from the hardness of deciding quadratic residuosity modulo composite numbers. In the ROM, the Fiat-Shamir transform [FS87] can be used to turn a Σ-protocol[3][CDS94] into a NIZK proof, which yields more efficient proofs for many practically relevant languages.

We will focus on definitions in the ROM and allow our algorithms to have access to a hash function H, which will be modelled as a random oracle. Definitions are partially taken from [FKMV12; Fis05].

Definition 32 (Non-Interactive Zero-Knowledge (NIZK) Proofs)**.** *A non-interactive zero-knowledge proof system for an NP language $\mathcal{L}$ defined by an efficiently verifiable binary relation $\mathcal{R}$ via $x \in \mathcal{L} \Leftrightarrow \exists w$ s.t. $(x, w) \in \mathcal{R}$ consists of the following functions, which have access to a hash function H:*

- $\pi \leftarrow \mathsf{Prove}^H(x, w)$*: The proof algorithm takes a statement x and a witness w. It outputs a proof π.*
- $b \leftarrow \mathsf{Vrfy}^H(x, \pi)$*: The verification algorithm takes as input a statement x and a proof π. It outputs a bit b.*

We require such a proof to be perfectly complete, computationally sound *and* zero-knowledge.

Definition 33 (Perfect Completeness)**.** *A proof system* $(\mathsf{Prove}, \mathsf{Vrfy})$ *is perfectly complete, if for any $(x, w) \in \mathcal{R}$, any hash function H, it holds*

$$\Pr\left[\mathsf{Vrfy}^H(x, \mathsf{Prove}^H(x, w)) = 1\right] = 1.$$

[2] Otherwise our assumption on the computational hierarchy would collapse, as in [GO94] it is shown that if a NIZK proof system exists for language $\mathcal{L}$, then $\mathcal{L}$ is in the computational class BPP If we had a NIZK scheme that works for all of NP this would imply NP $\subset$ BPP.

[3] A Σ-protocol is essentially a three-round/two-message zero-knowledge protocol.

Definition 34 (Computational Soundness)**.** *We say that a proof system* $(\mathsf{Prove}, \mathsf{Vrfy})$ *has computational soundness if for all* $\lambda \in \mathbb{N}$*, every collision resistant hash function* H*, every* $x \notin \mathcal{L}$ *and PPT adversaries* $\mathcal{A}$*, it holds*

$$\Pr\left[\ \mathsf{Vrfy}^H(x,\pi) = 1 : \pi \leftarrow \mathcal{A}^H(1^\lambda, x)\ \right] = \mathsf{negl}(\lambda).$$

Definition 35 (Zero-Knowledge)**.** *We say that a proof system* $(\mathsf{Prove}, \mathsf{Vrfy})$ *is zero-knowledge in the random oracle model, if there exists a PPT simulator* S *such that for all PPT distinguishers* $\mathcal{D}$ *the following distributions are computationally indistinguishable:*

- *Let* H *be a random oracle, set* $\pi_0 = \emptyset$*,* $\delta_0 = 1^\lambda$*. Repeat for* $i = 1, \ldots, n$ *until* $\mathcal{D}$ *stops:* $(x_i, w_i, \delta_i) \leftarrow \mathcal{D}^H(i, \pi_{i-1}, \delta_{i-1})$*, where* $\pi_i \leftarrow \mathsf{Prove}^H(x_i, w_i)$ *if* $(x_i, w_i) \in \mathcal{R}$ *or* $\pi \leftarrow \perp$ *otherwise. Output* $\mathcal{D}$*'s final output.*
- *Let* $(H_0, \tau_0) \leftarrow S(0, 1^\lambda)$*, set* $\pi_0 = \emptyset$*,* $\delta_0 = 1^\lambda$*. Repeat for* $i = 1, \ldots, n$ *until* $\mathcal{D}$ *stops:* $(x_i, w_i, \delta_i) \leftarrow \mathcal{D}^{H_{i-1}}(i, \pi_{i-1}, \delta_{i-1})$*, where* $(H_i, \pi_i, \tau_i) \leftarrow S(i, x_i, \tau_{i-1}, \texttt{YES})$ *if* $(x_i, w_i) \in \mathcal{R}$ *or* $(H_i, \pi_i, \tau_i) \leftarrow S(i, x_i, \tau_{i-1}, \texttt{NO})$ *otherwise. Output* $\mathcal{D}$*'s final output.*

The fact that the Fiat-Shamir transform yields such a NIZK proof scheme when starting with a Σ-protocol has been formally treated e.g. in [BR93; FKMV12].

Remark. In some parts of the literature, such a scheme is actually called a *Non-Interactive Zero-Knowledge* ***Argument***, as some authors require a **Proof** to have *Perfect Soundness*, where the probability of accepting a proof for a statement $x \notin \mathcal{L}$ is **zero** instead of negligible. Unless explicitly specified, for this work a NIZK proof is considered to be only computationally sound, which is in line with the nomenclature in [BR93; FKMV12].

2.11.3 Non-Interactive Zero-Knowledge Proofs of Knowledge

A *proof of knowledge* represents a strengthening of the soundness guarantee. In a proof of knowledge, intuitively, the prover wants to convince the verifier not only, that the statement is true, but also, that the prover also knows a witness for this statement. Formally, we want there to be an *extractor* algorithm, which allows our reduction proofs to obtain a valid witness for x from interaction[4] with a prover that can convincingly prove $x \in \mathcal{L}$. The intuitive argument is then, that the prover could have in the same way computed the witness and thus "knows" a witness already.

Definition 36 (Non-Interactive Zero-Knowledge Proofs of Knowledge (NIZKPoK))**.** *A non-interactive zero-knowledge proof of knowledge for an NP language* $\mathcal{L}$ *defined by an efficiently verifiable binary relation* $\mathcal{R}$ *via* $x \in \mathcal{L} \Leftrightarrow \exists w$ *s.t.* $(x, w) \in \mathcal{R}$ *consists of two algorithms* $(\mathsf{Prove}^H, \mathsf{Vrfy}^H)$ *as in Def. 32, which should fulfill perfect completeness, zero-knowledge and weak simulation extractability.*

Definition 37 (Weak Simulation Extractability)**.** *A proof system* $(\mathsf{Prove}^H, \mathsf{Vrfy}^H)$ *for language* $\mathcal{L}$ *with witness relation* $\mathcal{R}$ *is weakly simulation extractable if there is a negligible extraction error* μ *such that the following holds: Let* S *be the simulator from the zero-knowledge definition, then for every PPT algorithm* $\mathcal{A}$*, there exists an efficient (expected polynomial time) extractor algorithm* $\mathcal{E}_\mathcal{A}$ *with full black-box access to* $\mathcal{A}$ *including the power to rewind, such that the following holds:*
We define the following algorithm $(x^*, \pi^*, H_n, Q_\mathcal{A}, \mathcal{T}_\mathcal{A}) \leftarrow \mathsf{Sim}_{\mathcal{A},S}(1^\lambda; \rho)$*:*

[4] Usually, this includes the power to rewind.

- *We set $\mathcal{A}$'s random tape to ρ.*
- *Let $(H_0, \tau_0) \leftarrow S(0, 1^\lambda)$, set $\pi_0 = \emptyset$, $\delta_0 = 1^\lambda$. Repeat for $i = 1, \ldots, n$ until $\mathcal{A}$ stops: $(x_i, w_i, \delta_i) \leftarrow \mathcal{A}^{H_{i-1}}(i, \pi_{i-1}, \delta_{i-1}; \rho)$, where $(H_i, \pi_i, \tau_i) \leftarrow S(i, x_i, \tau_{i-1}, \texttt{YES})$ if $(x_i, w_i) \in \mathcal{R}$ or $(H_i, \pi_i, \tau_i) \leftarrow S(i, x_i, \tau_{i-1}, \texttt{NO})$ otherwise.*
- *Let (x^*, π^*) be $\mathcal{A}$'s final output, $Q_\mathcal{A}$ be the queries that $\mathcal{A}$ made to oracles H_i and $\mathcal{T}_\mathcal{A} = (x_i, \pi_i)_{i \in [n]}$ be the transcript of proofs provided to $\mathcal{A}$.*

Then let

$$\begin{aligned}\mathsf{acc} = \Pr[(x^*, \pi^*, H_n, Q_\mathcal{A}, \mathcal{T}_\mathcal{A}) \leftarrow \mathsf{Sim}_{\mathcal{A},S}(1^\lambda; \rho) : \\ (x^*, \pi^*) \notin \mathcal{T}_\mathcal{A} \wedge \mathsf{Vrfy}^{H_n}(x^*, \pi^*)] \\ \mathsf{ext} = \Pr[(x^*, \pi^*, H_n, Q_\mathcal{A}, \mathcal{T}_\mathcal{A}) \leftarrow \mathsf{Sim}_{\mathcal{A},S}(1^\lambda; \rho); w^* \leftarrow \mathcal{E}_\mathcal{A}(x^*, \pi^*, Q_\mathcal{A}, \mathcal{T}_\mathcal{A}; \rho) : \\ (x^*, \pi^*) \notin \mathcal{T}_\mathcal{A} \wedge (x^*, w^*) \in \mathcal{R}]\end{aligned}$$

Then, there exists a constant $d > 0$ and a polynomial p such that whenever $\mathsf{acc} \geq \mu$, it holds $\mathsf{ext} \geq \frac{1}{p}(\mathsf{acc} - \mu)^d$.

For context: It was shown in [FKMV12] that the Fiat-Shamir transform of a Σ-protocol is already weakly simulation extractable and therefore a NIZKPoK, under the condition that the underlying Σ-protocol has an extra property called *quasi unique responses*[5].

2.11.4 Online-Extractability for NIZKPoKs

In some scenarios, we can not assume that we can fully access and rewind adversarial code. In fact, if we are to model real-life online communication, we usually only get access to other parties' outputs, not their code and we have to assume that any party is stateful and will remember prior communication. In this scenario, we ideally want to be able to run the extractor only on a given message-proof pair. This is exactly what is captured by the notion of *online-extractable* NIZKPoKs.

An alternative compiler from Σ-protocols to NIZK proofs, which achieves online extractability is the Fischlin-transform [Fis05]. The downsides of this transform is that it leads to bigger proofs than the Fiat-Shamir-transform and achieves only computational completeness. We capture this in our definition.

Definition 38 (Online-Extractable Non-Interactive Zero-Knowledge Proofs of Knowledge)**.** *An online-extractable non-interactive zero-knowledge proof of knowledge for an NP language $\mathcal{L}$ defined by an efficiently verifiable binary relation $\mathcal{R}$ via $x \in \mathcal{L} \Leftrightarrow \exists w$ s.t. $(x, w) \in \mathcal{R}$ consists of two algorithms $(\mathsf{Prove}^H, \mathsf{Vrfy}^H)$ as in Def. 32, which should fulfill computational completeness, zero-knowledge and full simulation extractability.*

Definition 39 (Computational Completeness)**.** *A proof system $(\mathsf{Prove}^H, \mathsf{Vrfy}^H)$ is computationally complete, if for any $(x, w) \in \mathcal{R}$, any hash function H, it holds*

$$\Pr\left[\mathsf{Vrfy}^H(x, \mathsf{Prove}^H(x, w)) = 1\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 40 (Full Simulation Extractability)**.** *A proof system $(\mathsf{Prove}^H, \mathsf{Vrfy}^H)$ is fully simulation extractable (or: online extractable) if the following holds: There exists a PPT extractor $\mathcal{E}$, such that for every PPT algorithm $\mathcal{A}$ and the simulator S from the zero-knowledge definition, it holds:*

[5] This means that in the three-move proof system, it should be hard to find a statement x, first message α, second message β and third round messages γ, γ' such that both (α, β, γ) and (α, β, γ') are accepting transcripts for x. The authors argue that this is a mild assumption.

Let $(H_0, \tau_0) \leftarrow S(0, 1^\lambda)$*, set* $\pi_0 = \emptyset$*,* $\delta_0 = 1^\lambda$*. Repeat for* $i = 1, \ldots, n$ *until* $\mathcal{A}$ *stops:* $(x_i, w_i, \delta_i) \leftarrow \mathcal{A}^{H_{i-1}}(i, \pi_{i-1}, \delta_{i-1})$*, where* $(H_i, \pi_i, \tau_i) \leftarrow S(i, x_i, \tau_{i-1}, \texttt{YES})$ *if* $(x, w_i) \in \mathcal{R}$ *or* $(H_i, \pi_i, \tau_i) \leftarrow S(i, x_i, \tau_{i-1}, \texttt{NO})$ *otherwise. Let* (x^*, π^*) *be* $\mathcal{A}$*'s final output and* $Q_\mathcal{A}$ *be the queries that* $\mathcal{A}$ *made to oracles* H_i*. Let* $w^* \leftarrow \mathcal{E}(x^*, \pi^*, Q_\mathcal{A})$*. Then, if* $(x^*, \pi^*) \neq (x_i, \pi_i)$ *for all* $i \in [n]$*,*

$$Pr\left[(x^*, w^*) \notin \mathcal{R} \wedge \mathsf{Vrfy}^{H_n}(x^*, \pi^*) = 1\right] \leq \mathsf{negl}(\lambda).$$

For context: The Fischlin transform also has some additional requirements on the Σ-protocol for the transformation to work. Specifically, the Σ protocol needs to have quasi-unique responses and their second message must be a uniform challenge. We are actually only interested in this work in using the Fischlin transform on a specific Σ-protocol, the Schnorr proof, which fulfills these prerequesites. We will detail this scheme next.

2.11.5 Noteworthy Non-Interactive Zero-Knowledge Proofs

2.11.5.1 Schnorr's Proof of Knowledge for Discrete Log

We define the discrete logarithm problem due to Diffie and Hellman [DH76] for completeness:

Definition 41 (Computational Discrete logarithm)**.** *The discrete logarithm assumption states that the probability*

$$\Pr\left[\mathcal{A}(g, g^x) = x : x \leftarrow\!\!\$\ \mathbb{Z}_p\right]$$

is negligible for polynomial adversaries $\mathcal{A}$ *and generators* $g \in \mathbb{G}$ *of a group* $\mathbb{G}$ *of prime order* p*.*

We recall the three-message proof of knowledge for discrete logarithms due to Schnorr [Sch90] for the language induced by the relation $\mathcal{R} = \{(g^x, x) : x \in \mathbb{Z}_p\}$ where g is the generator of a group $\mathbb{G}$ of prime order p.

It consists of a proof algorithm $\mathsf{Schnorr.P}$ and a verification algorithm $\mathsf{Schnorr.V}$. Its proof algorithm is defined on input (h, x) for $h \in \mathbb{G}$ and $x \in \mathbb{Z}_p$.

$\mathsf{Schnorr.P}(h = g^x, x)$		$\mathsf{Schnorr.V}(h)$
$r \leftarrow\!\!\$\ \mathbb{Z}_p$		
$u \leftarrow g^r$		
	$\xrightarrow{u}$	
		$c \leftarrow\!\!\$\ \mathbb{Z}_p$
	$\xleftarrow{c}$	
$z \leftarrow r + cx$		
	$\xrightarrow{z}$	
		check whether $g^z = uh^c$. If so, return 1, else 0.

Table 2.1: Proof of Knowledge for Discrete Log

Fischlin [Fis05] discusses that this proof scheme fulfills all requirements for the Fischlin transformation to be applied. That means, that there exists a non-interactive online-extractable version of this zero-knowledge proof of knowledge for the relation $\mathcal{R} = \{(g^x, x) : x \in \mathbb{Z}_p\}$. In the following, we will refer to these transformed algorithms with the syntax $\pi \leftarrow \mathsf{Schnorr.Prove}(h, x)$ and $b \leftarrow \mathsf{Schnorr.Valid}(h, \pi)$.

2.11.5.2 Bulletproofs

Bulletproofs introduced by Bünz et al. [Bün+18] are a particularly efficient zero-knowledge proof of knowledge system, which is uniquely compatible with Pedersen commitments [Ped92] (see § 2.10.3.2). They allow to prove statements about the messages contained in Pedersen commitments.

The definitions in [Bün+18] are given with respect to interactive zero-knowledge proofs of knowledge in the CRS model, but they explain how a NIZKPoK proof in the ROM in the sense of Def. 36 can be obtained by applying the Fiat-Shamir transform [FS87] and retrieving the CRS from a hash of a fixed seed using the RO itself. In this thesis, we will always assume the CRS for bulletproofs to be instantiated in such a way and therefore only briefly mention the results that [Bün+18] achieve for interactive proofs in the CRS model without formally defining them.

Let $\mathsf{COM} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Vrfy})$ be the Pedersen commitment scheme in a group $\mathbb{G}$ of prime order p.

Lemma 2 ([Bün+18], Section 5.2, simplified)**.** *Given a relation $(x, w) \in \mathcal{R}$ with a membership test circuit for this relation of multiplicative complexity n.*

Under the discrete logarithm assumption, there exists a perfectly complete, weak simulation extractable and statistically zero-knowledge interactive proof for statements that are (partially) encoded in Pedersen commitments.

The proof consists of $2\lceil\log_2(n)\rceil + 8$ elements of $\mathbb{G}$ and 5 elements of $\mathbb{Z}_p$.

This construction can be turned into a NIZKPoK in the ROM with proof size $\mathcal{O}(\log_2(n))$ via the Fiat-Shamir transform. Specifically, for a fixed CRS there exists a NIZKPoK in the ROM for the Pedersen commitment scheme COM for the CRS-dependent language

$$((\mathsf{com}_1, \ldots, \mathsf{com}_\ell), x') \in \mathcal{L}_{\mathsf{CRS}} \Leftrightarrow \exists w, v_1, \ldots, v_\ell, \rho_1, \ldots, \rho_\ell \text{ s.t.}$$
$$\forall_{i\in[\ell]}(\mathsf{com}_i = \mathsf{COM.Commit}(\mathsf{CRS}, v_i; \rho_i)) \wedge ((v_1||v_2 \ldots||v_\ell||x'), w) \in \mathcal{R}.$$

Further, more explicitly, bulletproofs allow us to prove that the value embedded in a Pedersen commitment is in a certain *range*. Such a proof of knowledge, is also called a *range proof*.

Lemma 3 ([Bün+18], Section 4.2 and 4.4, simplified)**.** *There exists an interactive zero-knowledge range proof (P, V) for the Pedersen commitment scheme COM with range $[0, 2^n - 1]$ where the proofs π consist of $2\lceil\log_2(n)\rceil + 4$ group elements and 5 $\mathbb{Z}_p$ elements. This scheme can be transformed into a NIZKPoK in the ROM via Fiat-Shamir.*

Specifically, there is a NIZKPoK in the ROM for the language induced by the CRS-dependent relation $\mathcal{R}^{\mathsf{CRS}}_{range}$:

$$((\mathsf{com}, n), (x, \rho)) \in \mathcal{R}^{\mathsf{CRS}}_{range} \Leftrightarrow \mathsf{com} = \mathsf{COM.Commit}(\mathsf{CRS}, x; \rho) \wedge 0 \leq x < 2^n$$

with proof size in $\mathcal{O}(\log_2(n))$.

This can be aggregated; proving for m commitments $(\mathsf{com}_i)_{i\in[m]}$ that each com_i commits to x_i with $x_i \in [0, 2^n - 1]$ has total proof size only $\mathcal{O}(\log_2(nm))$.

2.12 Somewhere Statistically/Perfectly Binding Hashing

Somewhere statistically binding (SSB) hashing is a cryptographic primitive that allows us to hash an input vector into a small digest such that the output is statistically binding at a specific, hidden position. That is, there should not be two different

databases that are hashed to the same digest, but disagree on the position where the hashing key was chosen to be binding – that is except with negligible probability over the choice of the hashing key. SSB hashing was first introduced by Hubacek and Wichs in [HW15], where a construction is given that relies on combining Merkle-style Hashing with Fully Homomorphic Encryption.

Okamoto, Pietrzak, Waters, and Wichs [OPWW15] introduce a strengthened notion that allows

- a *local opening*: Similarly as in commitments an opening can be created to show that the value in the database underlying a hash h at index i is indeed some claimed value D_i.
- and *perfect binding*: In an SPB scheme honestly created keys are *always* binding.

Several constructions were given, including ones from the hardness of assumptions such as DDH and DCR (decisional composite residuocity, proposed by Paillier [Pai99]).

We will use as our standard notion of SPB hashing the notion of somewhere perfectly binding hashing with private local opening as introduced in [BDHKS19] by Backes, Döttling, Hanzlik, Kluczniak, and Schneider. Their notion does not require the key to be chosen honestly for the binding to hold, but introduces a secret key shk created with the hashing key, which is needed to create openings to values in the underlying database. The authors argue that the DDH and DCR based constructions mentioned above can easily adapted to fullfill their notion, thus the primitive given here can be instantiated from the hardness of assumptions such as DDH or DCR.

Definition 42 (Somewhere Perfectly Binding Hashing (with private local opening), [BDHKS19])**.** *A somewhere perfectly binding (SPB) hashing scheme* SPB *is composed of the following algorithms:*

- $(\mathsf{hk}, \mathsf{shk}) \leftarrow \mathsf{Gen}(1^\lambda, n, i)$ *takes as input the security parameter* λ, $n \in \mathbb{N}$ *and an index* $i \in [n]$*. It outputs a pair of hashing and secret keys* $(\mathsf{hk}, \mathsf{shk})$*.*
- $h \leftarrow \mathsf{Hash}(\mathsf{hk}, D)$ *takes as input a hashing key* hk *and a database* D *of size* n*. It deterministically outputs a hash value* h*.*
- $\tau \leftarrow \mathsf{Open}(\mathsf{hk}, \mathsf{shk}, D, i)$ *takes as input a hashing key* hk*, a secret key* shk*, a database* D *and an index* i*. It outputs a proof* τ*.*
- $b \leftarrow \mathsf{Verify}(\mathsf{hk}, h, i, x, \tau)$ *takes as input a hashing key* hk*, a hash value* h*, an index* i*, a value* x *and a proof* τ*. It deterministically outputs a bit* $b \in \{0, 1\}$*.*

We require that a SPB hashing scheme fulfills the following efficiency guarantees:

1. The hashing key hk and the proof τ are both of size $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$;
2. The Verify algorithm can be represented both by a circuit of size $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ and by a Turing machine of description size and runtime $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$.

Additionally, an SPB hashing scheme fulfills the following properties.

Definition 43 (Correctness)**.** *We say that a SPB hashing scheme is correct if for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all databases* D *of size* n*, all indices* $j, i \in [n]$ *we have that*

$$\Pr\left[1 \leftarrow \mathsf{Verify}(\mathsf{hk}, h, i, x, \tau) : \begin{array}{c}(\mathsf{hk}, \mathsf{shk}) \leftarrow \mathsf{Gen}(1^\lambda, n, j) \\ h \leftarrow \mathsf{Hash}(\mathsf{hk}, D) \\ \tau \leftarrow \mathsf{Open}(\mathsf{hk}, \mathsf{shk}, D, i)\end{array}\right] = 1.$$

Definition 44 (Somewhere Perfectly Binding)**.** *We say that a SPB hashing scheme is somewhere perfectly binding if for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all keys* hk*, all databases* D *of size* n*, all indices* $i \in [n]$*, all database values* x *and all proofs* τ *we have that*

$$\Pr\left[D_i = x : \begin{array}{c}h \leftarrow \mathsf{Hash}(\mathsf{hk}, D) \\ 1 \leftarrow \mathsf{Verify}(\mathsf{hk}, h, i, x, \tau)\end{array}\right] = 1.$$

Definition 45 (Index Hiding). *We say that a SPB hashing scheme is index hiding if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have that*

$$\left| \Pr \left[b \leftarrow \mathcal{A}_2(\mathsf{hk}, aux) : \begin{array}{c} (n, i_0, i_1, aux) \leftarrow \mathcal{A}_1(1^\lambda) \\ b \leftarrow\!\!\$\ \{0,1\} \\ (\mathsf{hk}, \mathsf{shk}) \leftarrow \mathsf{Gen}(1^\lambda, n, i_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

An alternative with a weaker binding guarantee is SSB. We now present the syntax of SSB which is almost identical to the one of SPB, but omits the private key. The binding guarantee of an SSB scheme only holds with respect to honestly generated keys, and there is a negligible probability that such a key fails to be binding.

Definition 46 (Somewhere Statistically Binding Hashing, [OPWW15]). *A somewhere statistically binding (with local opening) (SSB) scheme* SSB *is composed of the following algorithms*

- $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, n, i)$ *takes as input the security parameter λ, $n \in \mathbb{N}$ and an index $i \in [n]$. It outputs a hashing key* hk.
- $h \leftarrow \mathsf{Hash}(\mathsf{hk}, D)$ *which is the same as above.*
- $\tau \leftarrow \mathsf{Open}(\mathsf{hk}, D, i)$ *takes as input a hashing key* hk*, a database D and an index i. It outputs a proof τ.*
- $b \leftarrow \mathsf{Verify}(\mathsf{hk}, h, i, x, \tau)$*, which is the same as above.*

An SSB scheme also satisfies correctness, index hiding and the same efficiency requirements as above. These definitions can be straightforwardly adapted to SSB. Additionally, SSB must be somewhere statistically binding as defined below.

Definition 47 (Somewhere statistically binding w.r.t. Opening, [OPWW15]). *We say that a hashing key* hk *is statistically binding for index $i \in [n]$ if there are no values h,$x \neq x'$,π,π' such that $\mathsf{Vrfy}(\mathsf{hk}, h, i, x, \pi) = 1 = \mathsf{Vrfy}(\mathsf{hk}, h, i, x', \pi')$.*

An SSB hashing scheme is somewhere statistically binding if for all $\lambda \in \mathbb{N}$, all $n = \mathsf{poly}(\lambda)$, all indices $i \in [n]$ we have that

$$\Pr \left[\mathsf{hk} \text{ is statistically binding for index } i : \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, n, i) \right] \geq 1 - \mathsf{negl}(\lambda).$$

We remark that [OPWW15] also introduce and construct a perfectly binding version, which is achieved when the probability in Def. 47 is 1. This notion might be useful in some of our constructions to achieve lower security loss in our reductions, but we think it is conceptually easier for the reader to follow when we use only one type of SPB and one type of SSB in this thesis.

2.13 Witness Encryption

Witness encryption was introduced by Garg, Gentry, Sahai, and Waters [GGSW13]. It is a special type of encryption where a message is encrypted with respect to an NP statement. Decryption is only possible for someone holding a corresponding witness.

Definition 48 (Witness Encryption). *Let $\mathcal{L}$ be an* NP *language with relation $\mathcal{R}$. A witness encryption* WE *scheme for $\mathcal{L}$ is composed of the following algorithms:*

- $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, x, m)$ *takes as input the security parameter λ, a statement $x \in \mathcal{X}$ and a message m. It outputs a ciphertext* ct.
- $m \leftarrow \mathsf{Dec}(\mathsf{ct}, w)$ *takes as input a ciphertext* ct *and a witness $w \in \mathcal{W}$. It outputs a message m.*

We require such a scheme to be correct and soundness secure.

Definition 49 (Correctness). *We say that a WE is correct if for all $\lambda \in \mathbb{N}$, all $x \in \mathcal{L}$ and all messages m we have that*

$$\Pr\left[m \leftarrow \mathsf{Dec}(\mathsf{ct}, w) : \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, x, m)\right] = 1$$

where $w \in \mathcal{W}$ is such that $\mathcal{R}(x, w) = 1$.

The standard security definition for witness encryption is soundness security and we use the game-based version of this definition due to Bellare and Hoang [BH15].

Definition 50 (Soundness Security). *We say that a WE is soundness secure if for all $\lambda \in \mathbb{N}$, for all adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and for all $x \notin \mathcal{L}$ we have that*

$$\left|\Pr\left[b \leftarrow \mathcal{A}_2(\mathsf{ct}, aux) : \begin{array}{c}(m_0, m_1, aux) \leftarrow \mathcal{A}_1(1^\lambda) \\ b \leftarrow\!\!\$\ \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, x, m_b)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

WE can be constructed from multilinear maps [GGSW13] or $i\mathcal{O}$ [Gar+13].

Ciphertext Size. Let $\mathcal{C}_x$ be the verification circuit for a statement x and a relation $\mathcal{R}$, that is, $1 \leftarrow \mathcal{C}_x(w)$ if and only if $\mathcal{R}(x, w) = 1$. The cited WE schemes for a relation $\mathcal{R}$ achieve a ciphertext size $|\mathsf{ct}| \in \mathcal{O}\left(|\mathcal{C}_x| \cdot \mathsf{poly}(\lambda)\right)$ where $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, x, m)$.

For reference, Garg, Gentry, Halevi, and Zhandry [GGHZ14] implicitly define a WE scheme from multilinear maps for circuit satisfiability with the claimed ciphertext size. Ciphertext size scales with the number of gates in the underlying circuit.

2.14 Indistinguishability Obfuscation

Indistinguishability obfuscation ($i\mathcal{O}$) was introduced by Barak et al. [Bar+01] as an alternative to previous (unachievable) notions of obfuscation. It refers to an obfuscation where, given two functionally equivalent circuits of the same size, their obfuscated versions are computationally indistinguishable. Indistinguishability obfuscation is commonly regarded as the "best possible" obfuscation notion that we can hope to achieve, following a discussion by Goldwasser and Rothblum [GR07]. While early works rely on specific classes of functionalities to obfuscate, in [Gar+13] Garg et al. gave a candidate construction for $i\mathcal{O}$ for any polynomially sized circuits based on multilinear maps in 2013. A construction from more well-studied assumptions is given by Jain, Lin, and Sahai in [JLS21], where (subexponentially secure) $i\mathcal{O}$ is constructed from LWE, LPN, PRGs and SXDH. However despite these theoretical advancements concrete efficiency of $i\mathcal{O}$ remains elusive to date.

Definition 51 (Indistinguishability Obfuscation, [Gar+13]). *Let $\mathsf{iO}(1^\lambda, \mathcal{C})$ be a uniform PPT algorithm that takes as input a security parameter λ and a circuit $\mathcal{C}$ and outputs a circuit $\bar{\mathcal{C}}$. iO is called an $i\mathcal{O}$ obfuscator for a circuit family $\mathfrak{C}$ if the following properties hold.*

Definition 52 (Correctness). *We say that an $i\mathcal{O}$ obfuscator $i\mathcal{O}$ is correct if for all $\lambda \in \mathbb{N}$, all $\mathcal{C} \in \mathfrak{C}$, all inputs x we have that*

$$\Pr\left[\bar{\mathcal{C}}(x) = \mathcal{C}(x) : \bar{\mathcal{C}} \leftarrow \mathsf{iO}(1^\lambda, \mathcal{C})\right] = 1.$$

Definition 53 (Security). *We say that an $i\mathcal{O}$ obfuscator $i\mathcal{O}$ is secure if for all $\lambda \in \mathbb{N}$, all pairs $(\mathcal{C}_0, \mathcal{C}_1)$ such that $|\mathcal{C}_0| = |\mathcal{C}_1|$ and $\mathcal{C}_0(x) = \mathcal{C}_1(x)$ for all inputs x, and all PPT*

adversaries $\mathcal{A}$ we have that

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\bar{\mathcal{C}}) : \bar{\mathcal{C}} \leftarrow \mathsf{iO}(1^\lambda, \mathcal{C}_0)\right] - \Pr\left[1 \leftarrow \mathcal{A}(\bar{\mathcal{C}}) : \bar{\mathcal{C}} \leftarrow \mathsf{iO}(1^\lambda, \mathcal{C}_1)\right]\right| \leq \mathsf{negl}(\lambda).$$

2.14.1 Indistinguishability Obfuscation for Turing Machines

Besides indistinguishability obfuscation for circuits, the notion has been adapted to the Turing Machine computation model. Indistinguishability obfuscation for Turing Machines was constructed by Koppula, Lewko, and Waters [KLW15] as well as Garg and Srinivasan [GS18] from $i\mathcal{O}$ for circuits additionally assuming either One-Way functions and PRGs or somewhere statistically binding hash functions. Their constructions are succinct in the sense that the time required to obfuscate a machine grows only with its description size and does not depend directly on its run-time or space complexity, which would be the case if the machine was represented as a circuit first and then obfuscated with regular $i\mathcal{O}$ for circuits.

We recall the notion of $i\mathcal{O}$ for Turing Machines (almost verbatim) from [GS18]:

Definition 54 (Succinct Indistinguishability Obfuscator)**.** *A succinct indistinguishability obfuscator for a machine class $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of a uniform PPT machine $i\mathcal{O}\mathsf{M}$ as follows:*

- $\mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M\right)$*: $i\mathcal{O}\mathsf{M}$ takes as input the security parameter 1^λ, a description M of the Turing machine to obfuscate, and a unary encoded input length n and time bound t for M. The output of $i\mathcal{O}\mathsf{M}$ is a machine $\mathsf{ob}M$, which is an obfuscation of M corresponding to input length n and time bound t. $\mathsf{ob}M$ takes as input $x \in \{0,1\}^n$ and a run-time $t' \leq t$.*

The scheme should satisfy correctness, security and efficiency requirements as defined below:

Definition 55 (Correctness)**.** *A uniform PPT machine $i\mathcal{O}\mathsf{M}$ as above is correct, if for all security parameters $\lambda \in \mathbb{N}$, for all $M \in \mathcal{M}_\lambda$, for all inputs $x \in \{0,1\}^n$, time bounds t and $t' \leq t$, let y be the output of M on t' steps, then we have that:*

$$\Pr\left[\mathsf{ob}M\left(x, t'\right) = y : \mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M\right)\right] = 1.$$

Definition 56 (Security)**.** *A uniform PPT machine $i\mathcal{O}\mathsf{M}$ as above is secure if for any (not necessarily uniform) PPT distinguisher $\mathcal{A}$, there exists a negligible function negl such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, time bounds t, and pairs of machines $M_0, M_1 \in \mathcal{M}_\lambda$ with the same size and identical output behaviour (i.e. for all running times $t' \leq t$ and for all inputs x, $M_0(x) = M_1(x)$ when M_0 and M_1 are executed for time t'), we have that:*

$$\left|\Pr\left[1 \leftarrow \mathcal{A}\left(i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M_0\right)\right)\right] - \Pr\left[1 \leftarrow \mathcal{A}\left(i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M_1\right)\right)\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 57 (Efficiency and Succinctness)**.** *We require that the running time of $i\mathcal{O}\mathsf{M}$ as well as the length of its output, namely the obfuscated machine $\mathsf{ob}M$, is $\mathsf{poly}(|M|, \log t, n, \lambda)$. We also require that the obfuscated machine on input x and t' runs in time $\mathsf{poly}\left(|M|, t', n, \log t, \lambda\right)$.*

Chapter 3

Universal Ring Signatures

In this chapter, we will formally introduce the notion of Universal Ring Signatures (URS) and provide multiple constructions thereof in the standard model.

To remind the reader, the idea of a URS is to allow a user to produce a ring signature with respect to a set of public verification keys $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$, without requiring these keys to originate from the same signature scheme or adhere to a common structure. In other words, each vk_i can be a verification key from a *(possibly different) signature scheme*, and most importantly, *none of the verification keys is required to be compatible with known ring signature schemes.*

The chapter is taken almost verbatim from [BDW22], which is joint work with Pedro Branco and Nico Döttling published at Asiacrypt 2022.

3.1 Structure and Contributions

The structure of this chapter is as follows:

- In § 3.2 we will provide a technical outline detailing the high-level ideas of our constructions as well as comparisons to related work and a discussion of our results.
- In § 3.3 we introduce definitions that formalize the requirements for a universal ring signature as informally outlined before.
- Then in § 3.4, § 3.5 and § 3.6 we proceed to present three standard model URS constructions, offering different trade-offs between signature size, security guarantees and necessary assumptions. Our schemes are *fully* universal, in the sense that no assumptions on the structure of verification keys are made.

We now outline our constructions in a little more detail:

Compact URS using Complexity Leveraging. Our first construction in § 3.4 is a URS scheme with compact signatures, i.e., its signature size depends only logarithmically on the number of users in the ring and on the number of signature schemes. This scheme relies on superpolynomial hardness of standard assumptions. Specifically, we rely on superpolynomially secure signature schemes, a (polynomially secure) perfectly binding non-interactive commitment scheme, perfectly sound non-interactive witness-indistinguishability (NIWI) proof systems for NP and somewhere perfectly binding (SPB) hashing scheme. All of these primitives can be instantiated using standard hardness assumptions. We get the following theorem.

Informal Theorem 1 (Construction 1, Thms. 1 to 3)**.** *Assuming the existence of perfectly binding non-interactive commitment schemes, perfectly sound NIWI proof systems for* NP *and SPB hashing schemes (all three with polynomial security), there exists a universal ring signature scheme in the standard model with compact signatures under the condition that the underlying signature schemes are superpolynomially secure.*

While this construction provides the baseline for our investigation, it raises a natural question: is superpolynomial hardness inherently necessary to construct standard model universal ring signatures?

Non-Compact URS from Witness Encryption. We answer this negatively in § 3.5, where we provide a construction based on polynomial and falsifiable hardness assumptions. Concretely, we rely on the existence of a witness encryption (WE) scheme for NP, a perfectly sound NIWI proof system for NP, an SPB hashing scheme, and a pseudorandom function (PRF). In terms of compactness, the size of the signatures of this scheme depends linearly on the number of users in the ring. Further, this scheme fulfills a slightly relaxed notion of anonymity, which we call t-anonymity, requiring that at least t verification keys in the ring be honestly generated. The standard notion of anonymity corresponds to 2-anonymity.

Informal Theorem 2 (Construction 2, Thms. 4 to 6)**.** *Assuming the existence of a WE for NP, a perfectly sound NIWI proof system for NP, an SPB hashing scheme, and a PRF, there exists a (non-compact) universal ring signature scheme in the standard model with t-anonymity, where t is a parameter depending on the signature schemes involved.*

For all conceivable purposes, the parameter t here is a small constant. Concretely, t depends on the entropy κ of the honest verification keys involved. Asymptotically, any such key must have entropy at least $\kappa = \omega(\log(\lambda))$. Otherwise, it would be trivially insecure. Our only requirement on t will be that $t \cdot \kappa \geq \lambda$. In terms of concrete parameters, κ would have to be at least 50 bits (or else the underlying scheme would be trivially insecure). Setting $t = 3$ or $t = 4$ will be sufficient for this parameter choice.

This leaves open the question of compactness. Is perhaps any standard model URS necessarily non-compact?

Compact URS from Indistinguishability Obfuscation. We can also resolve this question negatively, yet under still a (potentially) stronger assumption: We provide a construction of a compact WE scheme from polynomial hardness assumptions for a special type of languages that we call (t, N) threshold conjunction languages, which together with the previous construction will imply a compact URS scheme from polynomial hardness assumptions.

A (t, N) threshold conjunction language is the set of statements $(x_1, \ldots, x_N)$ for which there are at least t valid statements x_i among them. The size of the ciphertexts we receive when encrypting under such a statement is compact in the sense that it only depends logarithmically on N. Our WE construction requires indistinguishability obfuscation ($i\mathcal{O}$), puncturable pseudorandom functions (PPRF), somewhere statistically binding (SSB) hashing schemes and (t, N)-linear secret sharing (LSS). We obtain the following theorem.

Informal Theorem 3 (Construction 3, Thms. 7 and 8)**.** *Assuming the existence of an $i\mathcal{O}$ for all circuits, a (non-compact) WE for NP, a PPRF, an SSB hashing scheme, and a (t, N)-LSS, there exists a compact WE scheme for (t, N) threshold conjunction languages, when $N - t \in \mathcal{O}(\log N)$.*

Combining the previous Construction 2 with this compact WE, we obtain our final URS construction. This URS construction achieves compact signatures. Specifically, we will show:

Informal Theorem 4 (Thm. 9)**.** *Assuming the existence of a compact WE for* $(N-1, N)$ *threshold conjunction languages, a perfectly sound NIWI proof system for* NP*, an SPB hashing scheme and a PPRF, there exists a compact universal ring signature scheme in the standard model with* t*-anonymity.*

Individual Contribution. The initial idea of making Universal Ring Signatures was given by Nico Döttling, who also largely wrote the introduction and technical outline of [BDW22]. Constructions were jointly discussed in research meetings. The main body was mostly written by Pedro Branco and myself, where contributions to writing were roughly equal, taking turns writing and correcting each other. My Master's thesis already borrowed the technique of combining SPB hashing and NIWI proofs to achieve efficient proofs of OR statements from [BDHKS19], so I took care of porting these techniques to the URS setting, while Pedro was more experienced with witness encryption and $i\mathcal{O}$ and kindly shared his knowledge with me. The definition of URS and Construction 1 based on complexity leveraging including underlying proofs was primarily written by me. Construction 2 based on WE was written with roughly equal contribution between Pedro and me, and I distinctly remember that the idea of using a PRF instead of a PRG originated from Nico after we received kind feedback from reviewers on a first draft of this paper pointing out that the yes- and no-instances of the PRG based language are potentially not properly separated. Construction 3 of a witness encryption for threshold conjunction languages was primarily written by Pedro. In the paper, an alternative construction of a witness encryption for threshold conjunction languages with polynomial security loss was written by Nico and Pedro, without my involvement, which I have included a short summary of for context in § 3.2.3.

3.2 Technical Overview

In this section, we provide a high-level overview of our URS constructions. We further provide context for their novelty by adding a discussion on folklore constructions in the ROM or using a CRS and the limitations of these idealized models in the context of universal ring signatures in § 3.2.5, as well as a comparison to related work in § 3.2.4.

Before presenting our constructions of URS, we briefly recall the ring signature scheme of Backes *et al.* [BDHKS19].

In the scheme of [BDHKS19] (which is itself based on [BKM06]), verification keys are composed of $\mathsf{VK}_i = (\mathsf{vk}_i, \mathsf{pk}_i)$ where vk_i is a verification key of a standard signature scheme Sig and pk_i is a public key of a public-key encryption (PKE) scheme that has pseudorandom ciphertexts[1].

To sign a message m with respect to a ring $R = \{\mathsf{VK}_i\}_{i\in[\ell]}$, the signer i first generates a signature $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$ and then encrypts σ using pk_i. That is, $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_i, \sigma)$. The signer then samples $\mathsf{ct}_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$. One crucial point is that, if the underlying PKE has pseudorandom ciphertexts, then we cannot distinguish well-formed ciphertexts from uniformly random strings. In particular, this means that ct_0 contains (computationally) no information about the public key under which it was encrypted.

The signer now proves that either ct_0 or ct_1 encrypts a valid signature under one of the verification keys in the ring using a non-interactive witness-indistinguishable (NIWI) proof system. If off-the-shelf NIWIs were used in this construction, the size of the proof would scale linearly with the size of the ring. This would lead to signatures

[1] Examples of such PKE schemes exist from the LWE or DDH assumption.

of size $\mathcal{O}(|R|\cdot\mathsf{poly}(\lambda))$, where λ is the security parameter. To circumvent this problem, [BDHKS19] employed a new strategy.

Compact NIWI Proofs. The main ingredient to compress the size of the NIWI proof is a somewhere perfectly binding (SPB) hashing scheme[2]. An SPB hashing scheme allows one to hash a database such that the hash perfectly binds to the database item at index i, while the hashing key hides the index i. The size of the hash depends only logarithmically on the database size. For more details, we refer to § 2.12 and the works [HW15; OPWW15; BDHKS19].

Given $\mathsf{ct}_0, \mathsf{ct}_1$, the signer can now use a NIWI proof system together with a somewhere perfectly binding (SPB) hashing scheme to create a compact proof π that either ct_0 or ct_1 encrypts a valid signature under one of the keys in the ring. The basic idea here is that instead of proving a statement over all verification keys in the ring, it is sufficient to prove a statement about just two SPB hashes. More concretely, to compute the proof π, the signer first generates an SPB pair of hashing key/secret key $(\mathsf{hk}_b, \mathsf{shk}_b) \leftarrow \mathsf{SPB.KeyGen}(1^\lambda, i)$ that binds to position i, for $b \in \{0,1\}$. Then, it hashes R into a digest $h_b \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_b, R)$ for $b \in \{0,1\}$. Finally, the signer proves that there exists an index i such that one of the two statements is true:

1. ct_0 encrypts a valid signature under vk_i and hk_0 binds to i;
2. ct_1 encrypts a valid signature under vk_i and hk_1 binds to i.

The signature is composed of $(\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \pi)$. By the efficiency requirements of SPB, the signature has size $\mathcal{O}(\log|R| \cdot \mathsf{poly}(\lambda))$.

Finally, to verify that a signature is valid, one just needs to deterministically recompute h_b as the hash of R under hk_b, for $b \in \{0,1\}$, and check that π is a valid proof.

Security. Let us retrace the proofs of unforgeability and anonymity provided in [BDHKS19]. To argue unforgeability, the security of the scheme is reduced to the security of the underlying signature scheme. To do this, the reduction receives a verification key vk_{i^*} from the challenger, creates the remaining verification keys vk_i, for $i \neq i^*$, and also the public keys pk_i for all $i \in [\ell]$. Importantly, the public keys pk_i are created such that the reduction knows the corresponding secret keys.

Upon receiving a (ring signature) forgery from the adversary, the reduction proceeds as follows:

1. Decrypt both ct_0 and ct_1 to obtain σ_0 and σ_1, respectively;
2. Check if either σ_0 or σ_1 is a valid signature under vk_{i^*}. If one of them is valid, the reduction outputs it as the forgery.

By the perfect binding of the SPB hashing and perfect soundness of the NIWI, the reduction outputs a valid forgery with non-negligible probability.

To prove anonymity, one relies on the witness-indistinguishability of the NIWI and the fact that the underlying PKE has pseudorandom ciphertexts. Concretely, given two honestly generated verification keys vk_{i_0} and vk_{i_1}, we can build a sequence of hybrids to prove that a signature created under vk_{i_0} is indistinguishable from a signature created under vk_{i_1}. The sequence of hybrids starts by replacing ct_1 with an encryption of a valid signature under vk_{i_1}, and this change goes unnoticed as a result of the pseudorandomness of the underlying PKE. Next, change the index in the

[2] In more detail an SPB scheme with private local opening is chosen, where a secret key is created alongside the hashing key to later assist in creating an opening to show that an element x was indeed at position i in the database underlying a given SPB hash. The advantage of this definition is that the binding property does not hinge on honest creation of the hashing key.

witness used to create the proof π from i_0 to i_1 using the witness-indistinguishability of the NIWI scheme.

3.2.1 Compact Universal Ring Signatures from Signatures with Superpolynomial Security

The construction of the Backes et al. scheme [BDHKS19] serves as the starting point of our first construction. Observe that the ring signature verification keys of the Backes et al. scheme have a special format: each verification key VK_i is composed of a standard verification key vk_i and a public key pk_i.

The public key pk_i, which can be chosen by the unforgeability reduction, is what enables this reduction to extract a valid forgery. In a URS, however, verification keys are not required to have any particular format. In particular, they are not required to include an independently chosen public key of a PKE. How can we facilitate the extraction of a forgery by an unforgeability reduction in the setting of URS?

Commitments instead of Ciphertexts. Our first observation is that the ciphertexts in the scheme of Backes et al. [BDHKS19] are never decrypted *in the actual scheme*. In effect, ciphertexts in this scheme act as extractable commitments. Thus, a natural approach is to rely on non-interactive commitments instead of ciphertexts in this construction. The main reason for using commitments instead of ciphertexts is that we can choose non-interactive commitment schemes that are *keyless*.

Using a commitment scheme CS, we can build a URS as follows: To sign a message m under a ring of users $R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell\}$ (where each vk_i is from a possibly different signature scheme), a signer first creates a signature $\sigma \leftarrow \mathsf{Sig.Sign}_i(\mathsf{sk}_i, m)$ using its signature scheme Sig_i. Then, it commits to $(\mathsf{com}_0, \gamma_0) \leftarrow \mathsf{CS.Commit}(1^\lambda, \sigma)$ and to $(\mathsf{com}_1, \gamma_1) \leftarrow \mathsf{CS.Commit}(1^\lambda, 0)$ (where γ_b is the opening information). Using SPB and NIWI exactly as before, the signer can create a compact proof π that one of the commitments hides a valid signature under one of the keys in R.

Anonymity follows by essentially the same argument as before, where the hiding property of the underlying commitment is used instead of the ciphertext pseudorandomness of the PKE in [BDHKS19].

Unforgeability from Superpolynomial Hardness. Next, we show how the unforgeability reduction can extract a valid forgery from the adversary. Assume that the hiding property of the commitment scheme CS holds against *polynomial-time* adversaries but that CS can be extracted in *superpolynomial-time*. We can then use *complexity leveraging* to prove the unforgeability of the scheme, assuming the underlying signature schemes are unforgeable against superpolynomial-time adversaries.

Concretely, given a PPT adversary $\mathcal{A}$ that breaks the unforgeability of our URS, we can construct a superpolynomial-time reduction against the unforgeability of one of the Sig_i. The reduction, after receiving a forgery $\Sigma^* = (\mathsf{com}_0^*, \mathsf{com}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \pi^*)$ by $\mathcal{A}$, opens both com_0 and com_1 by brute force to recover σ_0^* and σ_1^* respectively. Since the commitment scheme CS is extractable in superpolynomial time, the reduction succeeds in recovering σ_0^* and σ_1^*. Now, as before, the reduction tests if there is a $b \in \{0, 1\}$ such that $1 \leftarrow \mathsf{Sig.Verify}_i(\mathsf{vk}_i, m, \sigma_b^*)$ and outputs σ_b^* if that is the case.

3.2.2 Non-Compact Universal Ring Signatures from Witness Encryption

Considering both the construction of Backes et al. [BDHKS19] and our construction in the last paragraph, the question emerges of how one could possibly *efficiently* extract

a signature, even if we cannot embed an extraction trapdoor into the protocol by either utilizing a CRS or augmenting the verification keys.

Extracting via Witness Encryption. Our way out of this dilemma starts with the observation that by relying on a sufficiently strong tool, namely standard witness encryption (WE) [GGSW13], we can repurpose any *sufficiently cryptographic* object as a public key. In our case, these objects will be the verification keys of the honest parties.

Recall that a WE scheme for an NP language $\mathcal{L}$ (with relation $\mathcal{R}$) allows an encrypter to encrypt a message m with respect to a statement x. If $x \in \mathcal{L}$, then a party in possession of a witness w such that $\mathcal{R}(x, w) = 1$ can recover the encrypted m. But, if $x \notin \mathcal{L}$, then indistinguishability of encryptions holds. Currently, we have constructions of WE from indistinguishability obfuscation ($i\mathcal{O}$) [Gar+13] or multilinear maps [GGSW13], but WE is potentially a weaker assumption than either of these. Specifically, [GMM17] shows that black-box constructions of $i\mathcal{O}$ from WE are impossible.[3]

To use the security of WE, we need to craft a language $\mathcal{L}$ with distinct true and false statements, such that witnesses of true statements allow for decryption, whereas ciphertexts under false statements hide the encrypted message. Ideally, true and false statements should be indistinguishable. Our design-choice of true and false statements will be informed by the following consideration: Consider two distributions of (honest) verification keys, one where each honest vk is generated using truly random coins, and another one where each honest $\tilde{\mathsf{vk}}$ is generated using (possibly correlated) pseudorandom coins. It is clear that these distributions are computationally indistinguishable, however under the right circumstances we can also make them *statistically far*, meaning that one of them can serve as a distribution of true statements, while the other one will be the distribution of false statements.

More concretely, let PRG be a pseudorandom generator (PRG). We say that a verification key vk is *malformed* if it is created using random coins coming from a PRG, instead of using truly random coins. That is, for some seed s

$$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda; \mathsf{PRG}(s)).$$

Similarly, a *well-formed* key vk is created using truly random coins.

Consider the language $\mathcal{L}$ parameterized by ℓ distinct verification keys $\{\mathsf{vk}_i\}_{i\in[\ell]}$. The *yes* instances of $\mathcal{L}$ are the instances $\{\mathsf{vk}_i\}_{i\in[\ell]}$ where *all but one of the verification keys are malformed*. In other words, there exist $\{s_i\}_{i\in[\ell]\setminus\{i^*\}}$ with $i^* \in [\ell]$ such that for all $i \in [\ell] \setminus \{i^*\}$

$$(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_i(1^\lambda; \mathsf{PRG}(s_i)).$$

Looking ahead, the dichotomy between *all but one key are malformed* vs *at least two keys are well-formed* will enable us to prove unforgeability and anonymity respectively. In the former case, the statement under which the WE ciphertext is created is true and, thus, we will be able to decrypt it. In the latter case, the statement is false. Therefore, we can use the security of the WE scheme.

At first glance, this approach seems to work. However, there is a caveat: when the reduction wants verification keys to be well-formed, it might accidentally choose keys that are malformed. As an example, consider a signature scheme Sig whose verification keys have less min-entropy than the underlying PRG. Say the key generation algorithm $\mathsf{Sig.KeyGen}(1^\lambda, r)$ only uses the first $\lambda/3$ bits of r whereas the PRG

[3] More precisely, the authors rule out constructing $i\mathcal{O}$ from WE in an *extended black-box model*, which additionally allows the reduction to use the underlying primitive's own subroutines as callable gates inside its inputs.

seed has $\lambda/2$ bits of entropy. In other words, the distributions of well-formed keys and malformed keys might not be sufficiently statistically far. Then, there is a non-negligible probability that a key chosen from the well-formed distribution is actually malformed. We could assume that the underlying signature schemes have exponential security (e.g., verification keys have λ bits of min-entropy), but this would to some degree defeat the purpose of URS.

Replacing the PRG by a PRF. The solution for this problem is to use a pseudo-random function (PRF) instead of a PRG to sample malformed keys and to require a number $t \geq 2$ of honest keys in the ring, which depends on the minimum min-entropy that we expect our signature schemes to have. Instead of generating malformed keys individually, we now generate them in a correlated fashion: A set of keys $\{\mathsf{vk}_i\}$ is malformed iff a PRF key K exists such that

$$(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_i(1^\lambda; \mathsf{PRF}(K, i)).$$

Note that now, all malformed keys are correlated via the PRF key. This construction ensures that the distribution of t malformed keys has only λ bits of min-entropy, because as soon as we choose the PRF key, all malformed keys are fixed. On the other hand, when sampling t well-formed keys independently, the resulting distribution will have $t\kappa$ bits of min-entropy where κ is the min-entropy of each verification key. By setting $t\kappa > \lambda$, we ensure that the two distributions are statistically far apart.

This fact will allow us to prove t-anonymity by *making the number of honest keys in the ring just large enough.*

Given this, we redefine the language $\mathcal{L}$ in the following way: *yes* instances of $\mathcal{L}$ are the instances $\{\mathsf{vk}_i\}_{i\in[\ell]}$ where *all but one of the verification keys are malformed with respect to a single PRF key* K. In other words, there exists $K \in \{0,1\}^\lambda$ such that for all $i \in [\ell] \setminus \{i^*\}$ it holds

$$(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_i(1^\lambda; \mathsf{PRF}(K, i)).$$

The Scheme. Armed with a WE scheme WE for the language $\mathcal{L}$ described above, we now outline how we can construct a URS scheme.

The scheme is essentially the same as above except that we use the WE scheme for language $\mathcal{L}$ as a drop-in replacement for the commitment scheme.

To sign a message m with respect to the ring R, the signer encrypts a valid signature σ created using its own signing key. Then, it encrypts σ using WE under the statement $x = R$, that is, $\mathsf{ct}_0 \leftarrow \mathsf{WE.Enc}(1^\lambda, x, \sigma)$. Additionally, it creates the ciphertext $\mathsf{ct}_1 \leftarrow \mathsf{WE.Enc}(1^\lambda, x, 0)$. Finally, the signer can again use NIWI and SPB to prove compactly that one of the ciphertexts encrypts a valid signature.

We first analyze the size of the signature. In most known WE schemes, the ciphertext size is proportional to the size of the verification circuit for the language $\mathcal{L}$. Since the statement is of size $\mathcal{O}(|R| \cdot \mathsf{poly}(\lambda))$, then the ciphertexts output by WE are of size $\mathcal{O}(|R| \cdot \mathsf{poly}(\lambda))$. This implies that the signature is of size $\mathcal{O}(|R| \cdot \mathsf{poly}(\lambda))$.

Security. We now sketch the security proofs of the scheme. As mentioned before, we will set all but one key to be malformed in order to prove unforgeability. Whereas in the t-anonymity proof, we set none of the keys to be malformed (recall that t-anonymity requires that the challenge ring has at least t honestly generated verification keys).

To prove unforgeability, we design a reduction that sets all verification keys, but the challenge key vk_{i^*} to be malformed. That is, $\{\mathsf{vk}_i\}_{i\in[\ell]\setminus\{i^*\}}$ are malformedly created using a PRF key K. By the security of the PRF, the adversary is not able to distinguish the case where the verification keys $\{\mathsf{vk}_i\}_{i\in[\ell]\setminus\{i^*\}}$ are well-formed from the case when they are malformed.

The crucial observation now is that the reduction has a valid witness $w = K$ for the statement $x = R$ under which WE ciphertexts are encrypted. This means that, upon receiving a URS forgery

$$\Sigma^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \pi^*)$$

by the adversary, the reduction can use w to decrypt both ct_0^* and ct_1^*. An analysis identical as for the previous scheme shows us that, if Σ^* is a valid URS signature, then there is a non-negligible probability that one of ct_0^* and ct_1^* decrypts to a valid signature σ^* under vk_{i^*}.

In the t-anonymity proof, we set *none* of the verification keys to be malformed, from which the adversary chooses t of them, say, $\mathsf{vk}_{i_0}, \ldots, \mathsf{vk}_{i_{t-1}}$. If the parameters of the PRF are chosen properly, then there is a negligible probability that $x \in \mathcal{L}$. As explained above, since all t verification keys are sampled independently, it is unlikely that $t-1$ share correlations via a PRF key K. This is because the distribution of $t-1$ honestly generated keys has significantly more min-entropy than $t-1$ malformed keys. Thus, there will be *at least two well formed* verification keys in the challenge ring R^* with overwhelming probability. We conclude that WE encryptions of σ are indistinguishable from WE encryptions of 0 by the security of the WE.

Given this, we can easily build a sequence of hybrids in a similar fashion as for the previous schemes. That is, given two honestly generated verification keys $\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1}$ and a signature $\Sigma^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \pi^*)$ for a message m^* with respect to the ring R^* where $\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1} \in R^*$:

1. We first replace ct_1^* by an encryption of a valid signature σ' under vk_{i_1}. By the security of the underlying WE, this change is undetected by the adversary.
2. We switch witnesses from i_0 to i_1, using the witness-indistinguishability of the NIWI scheme.

3.2.3 Compact Universal Ring Signatures from Indistinguishability Obfuscation

At first glance, the techniques that we employed in the previous construction seem hopeless in our ultimate goal of building a compact URS from falsifiable hardness assumptions. On the one hand, for all known WE schemes that we know of, the size of the ciphertexts grows with the size of the statement. On the other hand, trying to reduce the size of the statement of the language $\mathcal{L}$ leads to challenges that undermine security.

The reason for this is that to be able to extract a valid forgery, the reduction needs to set up all verification keys but the challenge one in a *special mode*. In our case, the special mode is when keys are malformed. If the reduction sets just a few of them in this special mode, anonymity breaks down: An adversary could employ the same strategy as the unforgeability reduction to extract a signature from the challenge URS signature since, in the anonymity game, all but two verification keys may be adversarially chosen.

Given this state of affairs, it seems implausible that we can achieve a compact URS scheme just from general WE.

Our final contribution is to build a WE scheme for a special type of NP languages that we call *threshold conjunction languages*. A threshold conjunction language $\mathcal{L}'$ is

a language of the form

$$\mathcal{L}' = \{(x_1, \ldots, x_N) : \exists (x_{i_1}, \ldots, x_{i_{N-1}}) \text{ s.t. } x_{i_1} \in \mathcal{L} \wedge \cdots \wedge x_{i_{N-1}} \in \mathcal{L}\}.$$

In other words, given an instance $x = (x_1, \ldots, x_N)$, x is a yes instance of $\mathcal{L}'$ if *all but one* of the x_i are instances of $\mathcal{L}$.

Compact URS from Compact WE. Suppose that we have a *compact* WE scheme for threshold conjunction languages. That is, ciphertexts of such a scheme scale only logarithmically with N. Then, plugging this WE scheme into our construction from the previous section immediately yields a compact URS.

Compact Witness Encryption for Threshold Conjunction Languages. It remains to show how we can obtain such a scheme. For simplicity of this exposition, we focus on the case where we have N instances $x = (x_1, \ldots, x_N)$ and $x \in \mathcal{L}'$ iff $x_i \in \mathcal{L}$ for all $i \in [N]$. The case where all but one of the statements x_i must be true can be easily obtained by additionally using a secret sharing scheme.

The high-level idea of the construction is as follows: We build an obfuscated circuit $\bar{\mathcal{C}}$ that receives an index $i \in [N]$ and outputs non-compact WE ciphertexts $\mathsf{WE.Enc}(1^\lambda, x_i, r_i)$ for uniform $r_i \leftarrow\!\!\$\ \{0,1\}$.[4] The ciphertext of our new WE scheme for a message $m \in \{0,1\}$ is composed of $\bar{\mathcal{C}}$ and $c = m + \sum r_i$.

If one is in possession of witnesses for all statements x_i, then by the correctness of the underlying non-compact WE scheme, one can recover all r_i. On the other hand, if one of the statements x_{i^*} is false, then we can build a sequence of hybrids where we replace $\mathsf{WE.Enc}(1^\lambda, x_i, r_i)$ by an encryption of 0 and then replace c by a uniform value.

Although the idea seems to work at first glance, there is a critical issue: The scheme is not compact. This is because all the statements x_i must be hardwired into $\bar{\mathcal{C}}$, otherwise how does the circuit know under which statements it must encrypt each r_i? To circumvent this problem we again use a somewhere statistically binding (SSB) hashing scheme in a similar way as [HW15].[5] That is, the circuit only has a hash value $h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, \{x_1, \ldots, x_N\})$ hardwired. Now, when it receives (i, x_i, γ_i), it first checks if γ_i is a valid opening with respect to x_i, h. Since $\{x_1, \ldots, x_N\}$ is public, anyone can compute a valid opening

$$\gamma_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, \{x_1, \ldots, x_N\}, i)$$

for every $i \in [N]$.

Recall that the verification algorithm of an SSB hashing scheme can be implemented in size $\mathcal{O}(\log N \cdot \mathsf{poly}(\lambda))$. As a result, the circuit meets the desired efficiency bounds and has size $\mathcal{O}(\log N \cdot \mathsf{poly}(\lambda))$.

We thus obtain a WE scheme that outputs ciphertexts that depend only logarithmically on N.

[4] To make the circuit size independent of N, we use a pseudorandom function (PRF) to succinctly describe all the r_i. This PRF has to be puncturable in order to use the *puncturing technique* of Sahai and Waters [SW14].

[5] In this construction, we use the statistically binding variant. The reason we use this over an SPB scheme, is that our notion of SSB does not rely on a private key for generating openings. In the scheme we will give, it is crucial that openings are publicly computable, while we do no longer have to account for adversarially chosen hashing keys. This makes SSB suitable, where the somewhere binding guarantee hinges on the hashing keys being honestly generated.

How to Avoid the Exponential Security Loss of Current $i\mathcal{O}$ Schemes. We note that known indistinguishability obfuscation schemes typically suffer from a security loss that scales with the size of the obfuscated circuit's input domain [AJ15; BV15; BGLPT15]. This implies that the construction presented above suffers from an exponential security loss when we instantiate the $i\mathcal{O}$ scheme by any known construction since the circuit being obfuscated receives as input an index i, a statement x_i and an SSB proof γ_i and consequentially has an exponentially-sized domain. To address this, we refer the reader to our joint work [BDW22], where we present an alternative construction of compact witness encryption using $i\mathcal{O}$ for Turing machines, following the framework of Garg and Srinivasan [GS18] and laconic oblivious transfer techniques from Cho et al. [Cho+17], which keeps the obfuscation domain size polynomial and the security reduction efficient.

3.2.4 Related Work

Ring signatures have been extensively studied in the last two decades. Constructions in the random oracle model (ROM) include [RST01; AOS02; BGLS03; DKNS04; LPQ18]. Ring signatures in the CRS model were studied in [SW07; Boy07], where Boyen [Boy07] solves the interesting, but orthogonal, problem of how to include users in a ring whose public keys are not posted publicly by using a PKI structure. We can also find standard model constructions for ring signatures in works such as [BKM06; BDHKS19; PS19a].

All works presented above assume some form of structure on the verification keys. For example, the work of [RST01] assumes that ring verification keys are RSA keys or the work of [BKM06] assumes that ring verification keys are composed by a standard verification key and a uniformly random string.

The only exceptions we are aware of are the works of Abe, Ohkubo, and Suzuki as well as Goel, Green, Hall-Andersen, and Kaptchuk [AOS02; GGHK22]. In these works, ring signatures that support different signature schemes are presented. However, these works are only *somewhat universal* in the sense that there are signature schemes that are not compatible with their schemes. More precisely, the scheme of [AOS02] is compatible with *trapdoor-one-way* and *three-move* signature schemes. The scheme of [GGHK22] is compatible with certain sigma protocols. Any scheme outside of these classes is not compatible with their ring signature schemes. Moreover, these schemes are only secure in the ROM, whereas we work in the standard model. In essence, the focus of these works is different from ours as they sacrifice universality for efficiency. In this work, we take the opposite direction.

A construction of a universal primitive from $i\mathcal{O}$ has previously been given for a notion called signature aggregators by Hohenberger, Koppula, and Waters [HKW15]. This allows to combine signatures from different users using arbitrary signature schemes into one signature to succinctly store and verify. The application and techniques used are however different and can not be transferred to ring signatures trivially.

3.2.5 Discussion and Interpretation of our Results

Interpretation of our Results. Our results focus on the feasibility of *fully* universal ring signatures, but they also underscore a *negative result*: Namely, that no signature scheme can prevent users' keys from being misused in a ring signature, unless it is too weak to be considered secure in the first place.

Needless to say, the heavy cryptographic tools used in our constructions (e.g., witness encryption, indistinguishability obfuscation) are not currently practical. However, they provide the first rigorous foundation for URS in the standard model, and, crucially, without relying on trusted setup or heuristic models.

An open question we face nonetheless is whether we can construct (compact) URS from polynomial assumptions without relying on strong tools like witness encryption or $i\mathcal{O}$, or if perhaps a universal ring signing scheme can be shown to already imply such primitives.

A case can be made that URS are simpler to construct via Non-Interactive Zero-Knowledge proofs, if one is willing to employ either a common reference string (CRS) or the random oracle model (ROM). We will discuss such a construction and the potential problems arising when using CRS or ROM in an adversarial setting like ours.

Folklore URS Constructions via Non-Interactive Zero-Knowledge Proofs. Non-interactive zero-knowledge proofs of knowledge (NIZKPoK) offer a direct route to building URS. This type of proof system ensures that any convincing proof not only certifies the truth of a statement but also implies that the prover possesses a corresponding witness, specifically by making it possible for a reduction to extract the witness from the proof.

Given a ring R, a message m, and a commitment c, a NIZKPoK proof π can assert that c hides a signature σ such that (σ, m) verifies under some verification key $\mathsf{vk} \in R$. Extractability of the NIZKPoK ensures that the security can be reduced directly to the unforgeability of the underlying signatures.

There is a rich literature on constructing NIZKPoK proofs from standard assumptions in the CRS model [BFM88; FLS90; GOS06b; PS19b; BKM20; JJ21], or alternatively in the ROM [BR93]. If the goal was to construct a practically useful URS protocol for a set of compatible but independently designed signature schemes, then a construction using succinct NIZKPoK arguments would be preferable. In such a scenario, one could reasonably assume cooperation between users, including trust in a common reference string and in the involved signature schemes.

Challenges of Using CRS or ROM in our Setting. Our setting is different: users may be included in a ring adversarially, without their consent or cooperation. In this setting, using a CRS could give users who have been forced into a ring against their will a means of plausible deniability, e.g., by claiming that they do not trust the CRS that was used to generate a universal ring signature, as the party who generated such a CRS may also forge such a signature.

On the other hand, if we consider URS in the random oracle model, then the ROM-heuristic could cause issues. When protocols in the ROM are instantiated, we replace the random oracle with a concrete hash function H. As shown by Goldwasser and Kalai [GK03], this heuristic can lead to unsound proof systems if the underlying language already depends on this (concrete) hash function H.

This issue also comes up in the context of universal ring signatures, as one of the signature schemes could be chosen *depending on the hash function* H. Somewhat more concretely, assume we wanted to build a signature scheme Σ^* which makes a URS relying on a random oracle unsound, in the sense that if any verification key of Σ^* is used in a ring R, then universal ring signatures can be forged, while Σ^* is still EUF-CMA secure. We could achieve this by taking any EUF-CMA secure signature scheme Σ and modifying it to Σ^* by additionally including into the verification keys vk^* of Σ^* an obfuscated program $\mathcal{O}$ which lets anyone publicly generate URS of rings involving vk^*. Note that this obfuscated program $\mathcal{O}$ needs to *know* a succinct description of

the hash function H, but this is feasible as we assume H to be *instantiated*, rather than a random oracle. The same can, in fact, be argued for any fixed static common reference string CRS, i.e. the CRS can be hardwired into $\mathcal{O}$. Note that while for such a scheme the size of the verification key would increase, both generation and verification would remain essentially unchanged.

Looking ahead, if such a transformation from Σ to Σ^* was done starting relative to one of our standard-model secure URS, then Σ^* would be necessarily insecure. But for a URS whose unforgeability rests on the CRS model or the random oracle model, we would generally expect such a Σ^* to be unforgeable (once the CRS or the RO has been instantiated).

The bottom line of this discussion is that it seems hard to argue that URS constructions in the CRS model or the ROM would be robust against signature schemes which undermine the unforgeability of the URS by depending on the concrete CRS which is used or the concrete hash function which instantiates the Fiat-Shamir paradigm.

3.3 Defining Universal Ring Signatures

In this section, we introduce the definition of universal ring signatures[6]. We will compare our security definitions to the definitions in Bender, Katz, and Morselli [BKM06], who provide a classification of security definitions for standard ring signatures.

Definition 58 (Universal Ring Signature)**.** *A universal ring signature (URS) scheme* URS *is composed of a signing and a verification algorithm:*

- $\Sigma \leftarrow \mathsf{Sign}(1^\lambda, \mathsf{sk}_i, m, R, i, S)$ *takes as input a security parameter* 1^λ*, a signing key* sk_i*, a message* m*, a ring of keys* $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ *an index* $i \in [\ell]$ *and a list of signature schemes* $S = \{\mathsf{Sig}_i = (\mathsf{Sig.KeyGen}_i, \mathsf{Sig.Sign}_i, \mathsf{Sig.Verify}_i)\}_{i\in[M]}$*, where each* vk_j *is a public verification key under exactly one*[7] *of the schemes* Sig_i*. It outputs a signature* Σ*.*
- $b \leftarrow \mathsf{Verify}(\Sigma, m, R, S)$ *takes as input a signature* σ*, a message* m*, a ring of keys* R *and a list of signature schemes* S*. It outputs a bit* $b \in \{0, 1\}$*.*

We expect a URS to fulfill correctness, unforgeability and anonymity.

Definition 59 (Correctness)**.** *We say that a URS* $\mathsf{URS} = (\mathsf{Sign}, \mathsf{Verify})$ *is* correct *if for all* $\lambda \in \mathbb{N}$*, all* $\ell, M = \mathsf{poly}(\lambda)$*, all correct signature schemes* Sig'*, all* $j \in [\ell]$*, all messages* m *and all* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig}'.\mathsf{KeyGen}(1^\lambda)$*, we have that*

$$\Pr\left[1 \leftarrow \mathsf{Verify}\left(\mathsf{Sign}(1^\lambda, \mathsf{sk}, m, R, j, S), m, R, S\right)\right] = 1$$

for any $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ *such that* $\mathsf{vk}_j = \mathsf{vk}$ *and any* $S = \{\mathsf{Sig}_i\}_{i\in[M]}$ *such that* $\mathsf{Sig}' \in S$*. That is, the remaining elements in* R, S *may be arbitrarily chosen.*

We now define the unforgeability of a URS. A URS scheme should be compatible with any signature scheme. Hence, we would like to let the adversary choose signature schemes for the URS scheme. However, the adversary could choose an insecure signature scheme and, in this case, we cannot guarantee unforgeability. Hence, the experiment should provide a list of secure signature schemes and verification keys at the beginning of the experiment. The forgery given by the adversary must be

[6] We remark that the term universal ring signatures was also used by Tso [Tso13] to refer to a completely different property of ring signatures.

[7] In practice, keys/certificates are usually annoted with their respective schemes and we assume such a labelling here.

with respect to these verification keys. We note that, in the unforgeability definition for standard ring signatures in [BKM06] a similar situation happens: The forgery of the adversary must be with respect to verification keys created honestly and not with respect to maliciously chosen verification keys. Our definition is similar to the one of *unforgeability with respect to insider corruption* for standard ring signatures [BKM06], which is the strongest unforgeability definition.

Definition 60 (Unforgeability)**.** *Let $\mathcal{A}$ be an adversary. We denote by Ls a list of challenge signature schemes*

$$\mathsf{Ls} = \{\mathsf{Sig}_i = (\mathsf{Sig.KeyGen}_i, \mathsf{Sig.Sign}_i, \mathsf{Sig.Verify}_i)\}_{i\in[M]}.$$

Consider the following experiment, denoted by $\mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Unf}}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)$:

1. *The experiment provides Ls to $\mathcal{A}$.*
2. *The adversary outputs a list of indices $\{\mathsf{ind}_i\}_{i\in[\ell]}$.*
3. *For all $i \in [\ell]$, the experiment computes $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda)$ and outputs $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ to the adversary. Also it initialises a set $\mathcal{K} = \emptyset$ and remembers the indices ind_i.*
4. *The adversary may now make three types of requests[8]:*
   - $\texttt{Corrupt}(i)$*, which the experiment answers with the secret key sk_i. Also it adds vk_i to $\mathcal{K}$.*
   - $\texttt{URSSign}(m, \bar{R}, i, \bar{S})$ *takes as input an index $i \in [\ell]$, a message m, a ring of keys $\bar{R}$ (not necessarily contained in R) and a list of signature schemes $\bar{S}$. If $\mathsf{vk}_i \in \bar{R}$, we denote its position as i^*. If additionally $\mathsf{Sig}_{\mathsf{ind}_i} \in \bar{S}$, the experiment answers with $\Sigma \leftarrow \mathsf{URS.Sign}(1^\lambda, \mathsf{sk}_i, m, \bar{R}, i^*, \bar{S})$.*
   - $\texttt{Sign}(m, i)$ *takes as input an index $i \in [\ell]$ and a message m. The experiment answers with $\Sigma \leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_i}(1^\lambda, \mathsf{sk}_i, m)$.*
5. *$\mathcal{A}$ outputs $(\Sigma^*, m^*, R^*, S^*)$.*
6. *If $1 \leftarrow \mathsf{Verify}(\Sigma^*, m^*, R^*, S^*)$, $R^* \subseteq R \setminus \mathcal{K}$, $S^* \subseteq \mathsf{Ls}$ and the message m^* was never queried in a $\texttt{URSSign}$ or $\texttt{Sign}$ request, the experiment outputs 1.[9] Else, it outputs 0.*

We say that a URS $\mathsf{URS} = (\mathsf{Sign}, \mathsf{Verify})$ is unforgeable*, if for all $\lambda \in \mathbb{N}$, $M = \mathsf{poly}(\lambda)$, all lists of EUF-CMA secure signature schemes $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i\in[M]}$ and all PPT adversaries $\mathcal{A}$ we have that*

$$\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Unf}}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)\right] = \mathsf{negl}(\lambda).$$

In the anonymity experiment, the goal of the adversary is to guess which user created a given signature. We give a general definition called t-anonymity, which mandates that at least t honest keys in the anonymity set must be honestly chosen for anonymity to hold. The adversary may include at least t honest and additional maliciously chosen verification keys (potentially from insecure signature schemes) in a challenge ring. It should still be unable to determine which of the honest parties signed a given URS under that ring.

[8] Note that as the key generation algorithms are publicly available, the adversary may honestly generate key pairs itself. The corruption oracle simply serves to corrupt the initial honest keys. Arbitrary additional adversarially chosen keys can be included in ring signature queries, as we do not require $\bar{R} \subseteq R$.

[9] We can consider the stronger notion, where a forgery is valid, if no query of the form $\texttt{URSSign}(m^*, R^*, \cdot, \cdot)$ or $\texttt{Sign}(m^*||R^*, i)$ for $\mathsf{vk}_i \in R^*$ was made. This can be achieved by the standard trick of signing the message $(m^*||R^*)$ instead of m^* or a hash $H(m^*||R^*)$ thereof for compactness.

The case of 2-anonymity coincides with the definition of *anonymity against full key exposure* of [BKM06]. This is the strongest anonymity definition for ring signatures and is even known to imply unrepudiability, meaning that a member in the ring cannot prove that they did not sign the message [PS19a]. As it is the standard case, we will refer to 2-anonymity as *anonymity* throughout this work.

Definition 61 (t-Anonymity)**.** *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an adversary. We denote a list of challenge signature schemes by $\mathsf{Ls} = \{\mathsf{Sig}_i = (\mathsf{Sig.KeyGen}_i, \mathsf{Sig.Sign}_i, \mathsf{Sig.Verify}_i)\}_{i\in[M]}$.*

We define the t-anonymity experiment $\mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Anon}_t}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)$ as follows:

1. $(\{\mathsf{ind}_i\}_{i\in[\ell]}, aux_1) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{Ls})$.
2. *For all $i \in [\ell]$, the experiment computes $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda; \mathsf{coins}_i)$ with random coins coins_i and sets $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$.*
3. *$(m^*, R^* = (\mathsf{vk}'_1, \ldots, \mathsf{vk}'_p), S^* = (\mathsf{Sig}'_1, \ldots, \mathsf{Sig}'_q), (j_k)_{k\in[t]}, aux_2) \leftarrow \mathcal{A}_2(R, (\mathsf{coins}_1, \ldots, \mathsf{coins}_\ell), aux_1)$ where $\mathsf{vk}'_{j_k} \in R$ for $k \in [t]$ with indices l_k in R (i.e. $\mathsf{vk}'_{j_k} = \mathsf{vk}_{l_k}$). Additionally, the signature schemes corresponding to these public keys, $\mathsf{Sig}_{\mathsf{ind}_{l_k}}$, must be in the set S^*. If these conditions are violated, the experiment aborts.*
4. *$\Sigma^* \leftarrow \mathsf{URS.Sign}(1^\lambda, \mathsf{sk}_{l_k}, m^*, R^*, j_k, S^*)$ where $k \leftarrow\!\!\$\ [t]$.*
5. *$k' \leftarrow \mathcal{A}_3(\Sigma^*, aux_2)$.*
6. *If $k = k'$, then output 1. Else, output 0.*

We say that a URS $\mathsf{URS} = (\mathsf{Sign}, \mathsf{Verify})$ is t-anonymous*, if for all $\lambda \in \mathbb{N}$, all sizes $M = \mathsf{poly}(\lambda)$, all lists of signature schemes $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i\in[M]}$ and all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ we have that*

$$\left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Anon}_t}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)\right] - \frac{1}{t}\right| = \mathsf{negl}(\lambda).$$

Efficiency of URS. We remark that URS inherits the efficiency of the most inefficient signature scheme in the ring. For this reason, it is unlikely that we can construct a URS with good and practical parameters and efficiency. We consider a URS to be *compact* if its signature size grows only logarithmically in the number of keys in the ring as well as the number of signing schemes used.

3.4 Universal Ring Signature from Signature Schemes with Superpolynomial Security

In this section, we present a construction of URS that is based on signature schemes that are superpolynomially hard to forge. From this hardness, we can prove security of the URS scheme using complexity leveraging.

3.4.1 Construction

We start by presenting the construction of this URS scheme.

For simplicity, we assume, that there is an upper bound on the size of all descriptions of signature verification circuits. Also, for public keys $\mathsf{vk} \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, we assume that they are labeled with their respective schemes. That is, there is a function $\mathsf{tag}(.,.)$ which takes vk and a signature scheme Sig and outputs 1, iff the key vk was made under Sig, but 0 for any other signature verification scheme as input. $\mathsf{Sig.Vrfy}$ should only accept keys vk with the corresponding tag to Sig, that is $\mathsf{tag}(\mathsf{vk}, \mathsf{Sig}) = 1$.

In the scheme below, we use a non-interactive commitment scheme whose hiding property holds against PPT adversaries, but can be broken in superpolynomial time $T'(\lambda) \in \omega(\mathsf{poly}(\lambda))$ Additionally, we assume that all used signature schemes are unforgeable against adversaries running in $\mathcal{O}(T'(\lambda) \cdot \mathsf{poly}(\lambda))$. A signature of our URS for a message m includes a commitment to a signature of m in one of the underlying signature schemes. This will give our reduction, that runs in superpolynomial time, an advantage in the unforgeability experiment, where it may extract the commitments and provide a forgery against the underlying signature scheme. However, this opening strategy cannot be used by an adversary against anonymity, as they are running in polynomial time.

Construction 1. *Let:*

- CS *be a non-interactive commitment scheme such that the hiding property holds against polynomial-time adversaries but can be broken in superpolynomial-time* $T'(\lambda) \in \omega(\mathsf{poly}(\lambda))$.
- SPB *be a SPB hashing scheme;*
- $\mathcal{L}$ *be a language such that*

$$\mathcal{L} = \left\{ \begin{array}{c} (m, \mathsf{com}, \mathsf{hk}, h, \mathsf{rhk}, rh) : \exists(\mathsf{vk}, i, \mathsf{Sig.Verify}, \mathsf{ind}, \tau, \rho, \sigma, \gamma) \ s.t. \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{hk}, h, i, \mathsf{vk}, \tau) \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{rhk}, rh, \mathsf{ind}, \mathsf{Sig.Verify}, \rho) \\ 1 \leftarrow \mathsf{CS.Verify}(\mathsf{com}, \sigma, \gamma) \\ 1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}, m, \sigma) \end{array} \right\};$$

  where $\mathsf{Sig.Verify}$ *is a description of the verification algorithm of a signature scheme* Sig. *We assume that for all schemes,* $|\mathsf{Sig.Verify}|$ *is bounded by a polynomial* $\beta(\lambda)$.
- NIWI *be a NIWI scheme for the language*

$$\mathcal{L}_{\mathsf{OR}} = \left\{ \begin{array}{c} (m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1) : \\ \exists b \in \{0,1\} \ s.t. \ (m, \mathsf{com}_b, \mathsf{hk}_b, h_b, \mathsf{rhk}_b, rh_b) \in \mathcal{L} \end{array} \right\}.$$

We now describe our scheme in full detail.

$\mathsf{Sign}(1^\lambda, \mathsf{sk}_i, m, R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell), i, S = \{\mathsf{Sig}_i\}_{i \in [M]})$:

- *Determine an index* ind *such that* $\mathsf{tag}(\mathsf{vk_i}, \mathsf{Sig_{ind}})$. *Parse* $\mathsf{Sig_{ind}} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$. *Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i \in [M]}$ *to be the list of verification algorithms in* S.
- *Compute* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$.
- *Compute* $(\mathsf{hk}_b, \mathsf{shk}_b) \leftarrow \mathsf{SPB.Gen}(1^\lambda, \ell, i)$ *and* $h_b \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_b, R)$ *for* $b \in \{0,1\}$. *Also, compute the proof* $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0, \mathsf{shk}_0, R, i)$.
- *Compute* $(\mathsf{rhk}_b, \mathsf{rshk}_b) \leftarrow \mathsf{SPB.Gen}(1^\lambda, M, \mathsf{ind})$ *and* $rh_b \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_b, S')$ *for* $b \in \{0,1\}$. *Also, compute the proof* $\rho \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_0, \mathsf{rshk}_0, S', \mathsf{ind})$.
- *Compute* $(\mathsf{com}_0, \gamma_0) \leftarrow \mathsf{CS.Commit}(1^\lambda, \sigma)$ *and* $(\mathsf{com}_1, \gamma_1) \leftarrow \mathsf{CS.Commit}(1^\lambda, 0)$.
- *Set* $x = (m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1)$.
- *Set* $w = (\mathsf{vk}, i, \mathsf{Sig.Vrfy_{ind}}, \mathsf{ind}, \tau, \rho, \sigma, \gamma_0)$.
- *Compute the proof* $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$.
- *Output* $\Sigma = (\mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$.

$\mathsf{Verify}(\Sigma, m, R, S = \{\mathsf{Sig}_i\}_{i \in [M]})$:

- *Parse* Σ *as* $(\mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$. *Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i \in [M]}$ *to be the list of verification algorithms in* S.
- *Compute* $h_b \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_b, R)$ *for* $b \in \{0,1\}$.
- *Compute* $rh_b \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_b, S')$ *for* $b \in \{0,1\}$.

- *Set* $x = (m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1)$.
- *If* $1 \leftarrow \mathsf{NIWI.Verify}(x, \pi)$, *output* 1. *Else, output* 0.

We remark, that in order to verify a URS signature, we only require a description of the verification algorithms of the underlying signature schemes, not the whole description of the signature scheme. Therefore, we only include the verification algorithms in S', which is hashed down by SPB and provided to NIWI. This is to reduce size. Essentially, our verification algorithm URS.Vrfy could only take the list of signature verification algorithms S' as an input, but we state the full list of signature schemes to fit our more general definition.

Signature Size. A signature for a message m with respect to a ring R (of size ℓ) and a list of schemes S (of size M) is composed of $\Sigma = (\mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$. Both $\mathsf{com}_0, \mathsf{com}_1$ are of size $\mathcal{O}(\mathsf{poly}(\lambda))$ and independent of ℓ and M. The size of the hashing keys $\mathsf{hk}_0, \mathsf{hk}_1$, the proof τ and the circuit $\mathsf{SPB.Verify}(\mathsf{hk}, h, i, \mathsf{vk}, \tau)$ can be bounded by $\mathcal{O}(\log(\ell) \cdot \mathsf{poly}(\lambda))$. Analogously, $\mathsf{rhk}_0, \mathsf{rhk}_1$, ρ and the runtime of $\mathsf{SPB.Verify}(\mathsf{rhk}, rh, \mathsf{ind}, \mathsf{Sig.Verify}, \rho)$ are bounded by $\mathcal{O}(\log(M) \cdot \mathsf{poly}(\lambda))$. This holds, as we assumed, that we can bound $|\mathsf{Sig.Vrfy}|$ by a polynomial $\beta(\lambda)$ for all signature schemes Sig.

Given that, we conclude that the circuit that verifies the relation of language $\mathcal{L}$ has size at most $\mathcal{O}((\log(M) + \log(\ell)) \cdot \mathsf{poly}(\lambda))$. Hence, the proof π has size $\mathcal{O}((\log(M) + \log(\ell)) \cdot \mathsf{poly}(\lambda))$. We conclude that the total size of the signature is $\mathcal{O}((\log(M) + \log(\ell)) \cdot \mathsf{poly}(\lambda))$. Thus, it grows only logarithmically in the number of users in the ring and logarithmically in the number of signature schemes.

3.4.2 Proofs

We now show that the construction presented above fulfills the required properties for a URS due to Def. 58. We start by showing correctness. Then we proceed to prove unforgeability and anonymity. Our proof of unforgeability uses a superpolynomial-time reduction.

Theorem 1 (Correctness)**.** *The scheme presented in Construction 1 is correct, given that* NIWI *is perfectly complete and* SPB *and* CS *are correct.*

Proof. Let $\lambda \in \mathbb{N}$, $\ell, M = \mathsf{poly}(\lambda)$, $j \in [\ell]$, message m and a correct signature scheme Sig' be given. Let keys be constructed by $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig}'.\mathsf{KeyGen}(1^\lambda)$. Let a ring $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ be chosen with $\mathsf{vk}_j = \mathsf{vk}$ and $S = (\mathsf{Sig}_1, \ldots, \mathsf{Sig}_M)$ such that $\mathsf{Sig}' \in S$. Now we need to show, that

$$\Pr\left[1 \leftarrow \mathsf{Verify}\left(\mathsf{Sign}(1^\lambda, \mathsf{sk}, m, R, j, S), m, R, S\right)\right] = 1.$$

As SPB is deterministic and all other values are explicitly given, the input statement $(m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1)$ to NIWI.Vrfy is the same as what was originally input to NIWI.Prove. So by the perfect completeness of NIWI, we only need to show that $w \leftarrow (\mathsf{vk}, i, \mathsf{Sig.Vrfy}_{\mathsf{ind}}, \mathsf{ind}, \tau, \rho, \sigma, \gamma_0)$ was in fact a valid witness. The statement $1 \leftarrow \mathsf{CS.Verify}(\mathsf{com}_0, \sigma, \gamma_0)$ holds by construction and the correctness of CS. Also note, that by assumption there exists an index ind such that $\mathsf{Sig}_{\mathsf{ind}}$ is the scheme corresponding to vk, so such an index can be chosen in URS.Sign. The conditions in $\mathcal{L}_{\mathsf{OR}}$ are then fulfilled for the first disjunct by construction and using the correctness of SPB and $\mathsf{Sig}_{\mathsf{ind}}$. □

Theorem 2 (Unforgeability)**.** *We assume, that our non-interactive commitment scheme allows extraction in superpolynomial time* $T'(\lambda) \in \omega(\mathsf{poly}(\lambda))$*, but is secure against PPT adversaries. Assume that the challenge signature schemes* LS =

$\{\mathsf{Sig}_i\}_{i\in[M]}$ are unforgeable against adversaries running in time $T'(\lambda)\cdot\mathsf{poly}(\lambda)$. Then the scheme presented in Construction 1 is unforgeable against PPT adversaries, given that NIWI *is perfectly sound and* SPB *is somewhere perfectly binding.*

At a high-level, we will build a superpolynomial-time reduction that breaks unforgeability for the underlying signature scheme. The reduction, upon receiving the challenge URS signature $\Sigma^* = (\mathsf{com}_0^*, \mathsf{com}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \mathsf{rhk}_0^*, \mathsf{rhk}_1^*, \pi^*)$ from the adversary, opens the commitments com_0^* and com_1^* using brute force. Note that, since we allow the reduction to run in superpolynomial time, it will succeed in breaking the hiding property of the commiment scheme. Then, by the perfect soundess of the NIWI scheme, the reduction can extract a valid signature from either com_0 or com_1 with non-negligible probability and, thus, break the unforgeability of the signature scheme.

Proof. To prove the theorem, we show that if there exists a PPT adversary, that is able to win the unforgeability experiment of Def. 60, then we can build an algorithm that is able to forge a signature for one of the underlying Sig_i schemes in time $T'(\lambda)\cdot\mathsf{poly}(\lambda)$. More precisely, if $\mathcal{A}$ is able to break the unforgeability, then there exists an algorithm $\mathcal{B}$ that breaks the EUF-CMA security of one of the underlying signature schemes Sig_i. We guess which scheme it is in the beginning, incurring a polynomial security loss. In the following, let $\mathcal{C}$ be the challenger as in the EUF-CMA security game of the chosen signature scheme, detailed in Def. 3. We now describe $\mathcal{B}(\mathsf{Ls})$ in full detail:

1. $\mathcal{B}$ provides $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i\in[M]}$ to $\mathcal{A}$.
2. Upon receiving the indices $\{\mathsf{ind}_i\}_{i\in[\ell]}$ from $\mathcal{A}$, $\mathcal{B}$ guesses an index $i^* \leftarrow [\ell]$ uniformly at random. It creates $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda)$ for all $i \neq i^*$ and sets vk_{i^*} to be a verification key under $\mathsf{Sig}_{\mathsf{ind}_{i^*}}$ given by the challenger for EUF-CMA security of that scheme $\mathcal{C}$. It sends $R = \{\mathsf{vk}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.
3. $\mathcal{B}$ simulates the oracles `OCorrupt`, `URSign` and `OSign` in the following way:
   - `OCorrupt`(i): Whenever $\mathcal{A}$ sends a query i, $\mathcal{B}$ reveals sk_i if $i \neq i^*$. Otherwise, it aborts the protocol;
   - `URSign`$(m, \bar{R}, i, \bar{S})$: Whenever $\mathcal{A}$ sends a query $(m, \bar{R}, i, \bar{S})$, $\mathcal{B}$ proceeds as in the experiment, if $i \neq i^*$. If $i = i^*$, $\mathcal{B}$ checks whether $\mathsf{vk}_{i^*} \in \bar{R}$ and denotes its position as j^*. If additionally $\mathsf{Sig}_{\mathsf{ind}_{i^*}} \in \bar{S}$, $\mathcal{B}$ sends a query to $\mathcal{C}$ for message m. Upon receiving the signature σ for m, they compute a signature Σ as they would in $\mathsf{URS.Sign}(1^\lambda, \mathsf{sk}_{i^*}, m, \bar{R}, j^*, \bar{S})$, except that they use the σ they received instead of computing it by $\mathsf{Sig.Sign}_{\mathsf{ind}_{i^*}}(\mathsf{sk}_{i*}, m)$. This can be done without the knowledge of sk_{i^*}.
     $\mathcal{B}$ returns Σ to $\mathcal{A}$.
   - `OSign`(i, m): Whenever $\mathcal{A}$ sends a query (i, m) with $i \neq i^*$, $\mathcal{B}$ sends $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$. Otherwise, it sends a query m to $\mathcal{C}$ and, upon receiving a signature σ, it outputs σ to $\mathcal{A}$.
4. Upon receiving $\Sigma^* = (\mathsf{com}_0^*, \mathsf{com}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \mathsf{rhk}_0^*, \mathsf{rhk}_1^*, \pi^*)$, plus m^*, R^* and S^* from $\mathcal{A}$, $\mathcal{B}$ opens both commitments com_0 and com_1 to recover σ_0^* and σ_1^*, respectively. By assumption, this can be done in time $2T'(\lambda)$. If $1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}_{i^*}, m^*, \sigma_0^*)$, it outputs σ_0^*. Else if $1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}_{i^*}, m^*, \sigma_1^*)$, it outputs σ_1^*. Else, it aborts.

We now analyze the success probability of $\mathcal{B}$ in breaking the EUF-CMA property of Def. 3.

Before $\mathcal{A}$ outputs a potential forgery, unless they query to corrupt vk_{i^*}, $\mathcal{B}$ only differs from the experiment, in that it queries a signing oracle for vk_{i^*} instead of computing signatures itself. Therefore until such a query would be made, the index

i^* is uniform in the view of $\mathcal{A}$. The probability that $\mathcal{A}$ does not query $\mathsf{OCorrupt}$ on i^* is thus at least $1/\ell$. We now condition on this query not happening: Assume that Σ^* output by $\mathcal{A}$ is such that $1 \leftarrow \mathsf{Verify}(\Sigma^*, m^*, R^*, S^*)$, where $R^* = (\mathsf{vk}_{i_1}, \ldots, \mathsf{vk}_{i_{\ell'}})$.

Since NIWI is perfectly sound, then w.l.o.g, $(m^*, \mathsf{com}_0, \mathsf{hk}_0, h_0, \mathsf{rhk}_0, rh_0) \in \mathcal{L}$. This means there exists a tuple $(\mathsf{vk}', i', \mathsf{Sig.Vrfy}', \mathsf{ind}', \tau', \rho', \sigma', \gamma')$ such that it holds:

$$1 \leftarrow \mathsf{SPB.Verify}(\mathsf{hk}_0, h_0, i', \mathsf{vk}', \tau'), 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{rhk}_0, rh_0, \mathsf{ind}', \mathsf{Sig.Verify}', \rho'),$$
$$1 \leftarrow \mathsf{CS.Verify}(\mathsf{com}_0, \sigma', \gamma') \text{ and } 1 \leftarrow \mathsf{Sig.Verify}'(\mathsf{vk}', m^*, \sigma').$$

Thus, by the somewhere perfectly binding property of SPB we have that $\mathsf{vk}' = \mathsf{vk}_{i_{i'}}$ with probability 1. Moreover, since i^* is chosen uniformly at random from the point-of-view of $\mathcal{A}$, then $i^* = i_{i'}$ with probability at least $1/\ell$. Since we assumed that the vk are tagged with what scheme they correspond to, we can assume here that this is then a valid forgery under Sig'.

If this happens, then σ' is a valid signature for message m^* under vk_{i^*} and $\mathcal{B}$ is able to break the EUF-CMA of Sig_{i^*} with probability at least $\frac{1}{\ell^2} \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Unf}}\right]$.

Now, we need to analyse the time which $\mathcal{B}$ requires. Since $\mathcal{A}$ is PPT, and the answering of requests is possible in polynomial time as well, as all algorithms invoked run in $\mathsf{poly}(\lambda)$, the time spent until $\mathcal{A}$ outputs a forgery is in $\mathsf{poly}(\lambda)$. Then, $\mathcal{B}$ runs in a total time of $2T'(\lambda) + \mathsf{poly}(\lambda)$. This contradicts the assumption that Sig_{i^*} is EUF-CMA against adversaries running in time $\mathsf{poly}(\lambda) \cdot T'(\lambda)$. □

Theorem 3 (Anonymity)**.** *Assume that* SPB *is index hiding,* NIWI *is witness-indistinguishable and* CS *is computationally hiding. Then the scheme presented in Construction 1 is 2-anonymous following Definition 61.*

To prove the theorem above, we build a sequence of hybrids starting from the anonymity game where $b = 0$ and ending at a hybrid describing the game for $b = 1$. Let vk_{i_0} and vk_{i_1} be the challenge verification keys in the anonymity game and let $\Sigma = (m^*, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$ be the challenge signature build using vk_{i_0}. In the first hybrid, we change hk_1 and rhk_1 to be SPB hashing keys binding to index i_1. Next, we replace com_1 by a commitment of a valid signature under vk_{i_1}. In the next hybrid, we can replace the proof π by a new one computed using the new signature under vk_{i_1} (this change goes unnoticed by the witness indistinguishability of the NIWI). We can now replace com_0 by a commitment of a valid signature under vk_{i_1}. In the next step, we replace hk_0 and rhk_0 to be SPB hashing keys binding to index i_1 and, finally, compute π as the proof that com_0 is a commitment to a valid signature under vk_{i_1}, which hk_0 and rhk_0 bind to. Then we set com_1 to a commitment of 0 again and reach the game with $b = 1$.

Proof. Let $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i \in [M]}$ be a list of signature schemes. In the following we will only modify our response to $\mathcal{A}$ after they made their challenge query. That means, we have already received indices $\{\mathsf{ind}_i\}_{i \in \ell}$ from $\mathcal{A}$ and provided them with a ring R of verification keys as well as random coins. Let now $(i_0, i_1, m^*, R^*, S^*)$ be the challenge query of $\mathcal{A}$ in the game of Def. 61. We denote by j_0, j_1 the indices of $\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1}$ in R^* and by $\mathsf{ind}'_0, \mathsf{ind}'_1$ the indices of their corresponding signature schemes in S^*. Let S' be the list of signature verification algorithms in S^*. The proof of the theorem follows from the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This is the real anonymity experiment defined in Def. 61 where $b = 0$. That is, the challenger outputs $\Sigma = (m^*, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$ where

$$\begin{aligned}
\sigma_{i_0} &\leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_{i_0}}(\mathsf{sk}_{i_0}, m^*) \\
(\mathsf{hk}_a, \mathsf{shk}_a) &\leftarrow \mathsf{SPB.Gen}(1^\lambda, |R^*|, j_0) \text{ for } a \in \{0,1\} \\
\tau &\leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0, \mathsf{shk}_0, R^*, j_0) \\
(\mathsf{rhk}_a, \mathsf{rshk}_a) &\leftarrow \mathsf{SPB.Gen}(1^\lambda, |S'|, \mathsf{ind}'_0) \text{ for } a \in \{0,1\} \\
\rho &\leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_0, \mathsf{rshk}_0, S', \mathsf{ind}'_0) \\
(\mathsf{com}_0, \gamma_0) &\leftarrow \mathsf{CS.Commit}(1^\lambda, \sigma_{i_0}) \\
(\mathsf{com}_1, \gamma_1) &\leftarrow \mathsf{CS.Commit}(1^\lambda, 0) \\
\pi &\leftarrow \mathsf{NIWI.Prove}(x, w)
\end{aligned}$$

with statement $x = (m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1)$ and witness $w = (\mathsf{vk}, j_0, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_0}}, \mathsf{ind}'_0, \tau, \rho, \sigma_{i_0}, \gamma_0)$.

Hybrid $\mathcal{H}_1$. This hybrid is identical to the previous one except that it switches the index in key creation to $(\mathsf{hk}_1, \mathsf{shk}_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |R^*|, j_1)$. Additionally, it computes a new opening $\tau' \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_1, \mathsf{shk}_1, R^*, j_1)$.

Claim 1. *Assume that* SPB *is index hiding. Then hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable.*

Proof. Assume an adversary $\mathcal{A}$ has a non-negligible advantage in distinguishing $\mathcal{H}_0$ from $\mathcal{H}_1$. We build a reduction $\mathcal{R}$ against the index-hiding game as follows: $\mathcal{R}$ chooses $(|R^*|, j_0, j_1)$ as challenge and receives hk_1 from the challenger. $\mathcal{R}$ uses this key instead of computing $(\mathsf{hk}_1, \mathsf{shk}_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |R^*|, j_0)$ in an otherwise perfect simulation of $\mathcal{H}_0$. As τ' is not output in $\mathcal{H}_1$, from the view of $\mathcal{A}$, the output of $\mathcal{R}$ is identical to $\mathcal{H}_0$, if the challenge bit in the index hiding game was 0 and identical to $\mathcal{H}_1$ otherwise. Therefore, we can output whatever $\mathcal{A}$ outputs and receive the same advantage. □

Hybrid $\mathcal{H}_{1b}$. This hybrid is identical to the previous one except that it creates hashing keys $(\mathsf{rhk}_1, \mathsf{rshk}_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |S'|, \mathsf{ind}'_1)$. Additionally, it computes an opening $\rho' \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_1, \mathsf{rshk}_1, S', \mathsf{ind}'_1)$.

Claim 2. *Assume that* SPB *is index hiding. Then hybrids $\mathcal{H}_1$ and $\mathcal{H}_{1b}$ are indistinguishable.*

The proof is identical to the one for Claim 1.

Hybrid $\mathcal{H}_2$. This hybrid is identical to the previous one except that it uses $\sigma' \leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_{i_1}}(\mathsf{sk}_{i_1}, m^*)$ and $(\mathsf{com}_1, \gamma_1) \leftarrow \mathsf{CS.Commit}(1^\lambda, \sigma')$.

Claim 3. *Assume that* CS *is computationally hiding. Then hybrids $\mathcal{H}_{1b}$ and $\mathcal{H}_2$ are indistinguishable.*

Proof. Assume an adversary $\mathcal{A}$ has a non-negligible advantage in distinguishing $\mathcal{H}_{1b}$ from $\mathcal{H}_2$. We build a reduction $\mathcal{R}$ against the hiding property of CS as follows: $\mathcal{R}$ chooses $m_0 = 0$ and $m_1 = \sigma'$ as challenge in the hiding game. Then it receives com from the challenger and uses this commitment as com_1 instead of computing com_1 itself in an otherwise perfect simulation of $\mathcal{H}_{1b}$. As γ_1 is not needed in $\mathcal{H}_{1b}$ or $\mathcal{H}_2$, all further computations are possible and the output of $\mathcal{R}$ is identically distributed to $\mathcal{H}_{1b}$, if com commits to 0, and it is identically distributed to $\mathcal{H}_2$ otherwise. Therefore, we can output whatever $\mathcal{A}$ outputs and receive the same advantage. □

Hybrid $\mathcal{H}_3$. This hybrid is identical to the previous one except that it uses witness $w' = (\mathsf{vk}_{i_1}, j_1, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_1}}, \mathsf{ind}'_1, \tau', \rho', \sigma', \gamma_1)$ to compute $\pi \leftarrow \mathsf{NIWI.Prove}(x, w')$.

Claim 4. *Assume that* NIWI *is witness-indistinguishable. Then hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable.*

Proof. Assume an adversary $\mathcal{A}$' has a non-negligible advantage in distinguishing $\mathcal{H}_2$ from $\mathcal{H}_3$. We build a reduction $\mathcal{R}$ against the witness indistinguishability game as follows: $\mathcal{R}$ chooses (x, w, w') where $w = (\mathsf{vk}_{i_0}, j_0, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_0}}, \mathsf{ind}'_0, \tau, \rho, \sigma_{i_0}, \gamma_0)$ and $w' = (\mathsf{vk}_{i_1}, j_1, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_1}}, \mathsf{ind}'_1, \tau', \rho', \sigma', \gamma_1)$ as their challenge. Then it receives a proof π from the challenger that it uses instead of computing $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$ in an otherwise perfect simulation of $\mathcal{H}_2$. We remark that by our previous changes, the conditions for the second disjunct in $\mathcal{L}_{\mathsf{OR}}$ are now fulfilled by correctness of SPB, CS and Sig if the witness is w'. For the witness w, the first disjunct still holds, as no changes were made to its inputs. This means the statement-witness pairs were valid in all hybrids so far.

Now, clearly the output of $\mathcal{R}$ is identically distributed to $\mathcal{H}_2$, if the challenge bit in the witness indistuingishability game was 0 and to $\mathcal{H}_3$ otherwise. Therefore, we can output whatever $\mathcal{A}$ outputs and receive the same advantage. □

Hybrid $\mathcal{H}_4$. This hybrid is identical to the previous one except that it randomly chooses $\mathsf{com}_0 \leftarrow \mathsf{CS.Commit}(1^\lambda, 0)$.

Claim 5. *Assume that* CS *is computationally hiding. Then hybrids $\mathcal{H}_3$ and $\mathcal{H}_4$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 3.

Hybrid $\mathcal{H}_5$. This hybrid is identical to the previous one except that it computes $(\mathsf{hk}_0, \mathsf{shk}_0) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |R^*|, j_1)$, $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0, \mathsf{shk}_0, R^*, j_1)$, $(\mathsf{rhk}_0, \mathsf{rshk}_0) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |S'|, \mathsf{ind}'_1)$ and $\rho \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_0, \mathsf{rshk}_0, S', \mathsf{ind}'_1)$.

Claim 6. *Assume that* SPB *is index hiding. Then hybrids $\mathcal{H}_4$ and $\mathcal{H}_5$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 1.

Hybrid $\mathcal{H}_6$. This hybrid is identical to the previous one except that it uses $\sigma'' \leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_{i_1}}(\mathsf{sk}_{i_1}, m^*)$ and $(\mathsf{com}_0, \gamma_0) \leftarrow \mathsf{CS.Commit}(1^\lambda, \sigma'')$.

Claim 7. *Assume that* CS *is computationally hiding. Then hybrids $\mathcal{H}_5$ and $\mathcal{H}_6$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 3.

Hybrid $\mathcal{H}_7$. This hybrid is identical to the previous one except that it uses witness $w'' = (\mathsf{vk}_{i_1}, j_1, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_1}}, \mathsf{ind}'_1, \tau, \rho, \sigma'', \gamma_0)$ to compute $\pi \leftarrow \mathsf{NIWI.Prove}(x, w'')$.

Claim 8. *Assume that* NIWI *is witness-indistinguishable. Then hybrids $\mathcal{H}_6$ and $\mathcal{H}_7$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 4. We note, that now again, the first disjunct in $\mathcal{L}_{\mathsf{OR}}$ is fulfilled by the previous modifications.

Hybrid $\mathcal{H}_8$. This hybrid is identical to the previous one except that it randomly chooses $\mathsf{com}_1 \leftarrow \mathsf{CS.Commit}(1^\lambda, 0)$.

Claim 9. *Assume that* CS *is computationally hiding. Then hybrids $\mathcal{H}_7$ and $\mathcal{H}_8$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 3.

Since $\mathcal{H}_8$ is identical to the experiment with challenge bit $b = 1$ and indistinguishable from $\mathcal{H}_0$, we have therefore shown, that $\mathcal{A}$ can not have more than non-negligible advantage. □

3.5 Non-Compact Universal Ring Signature from Witness Encryption

In this section we present a URS scheme from falsifiable assumptions. The resulting URS has a signature size that scales with the size of the ring. We first present the construction. Then, we proceed to the analysis of the scheme.

3.5.1 Construction

We now present our construction for URS from WE.

Construction 2. *Let*

- $\mathsf{PRF} : \mathcal{K} \times [\ell] \to \{0,1\}^\lambda$ *be a PRF.*
- $\mathcal{L}'$ *be a language such that*

$$\mathcal{L}' = \left\{ \begin{array}{c} (\{\mathsf{vk}_i\}_{i\in[\ell]} : \exists \left(\{\mathsf{Sig}_{i_j}\}_{j\in[\ell-1]}, K\right) \ s.t. \\ r_{i_j} \leftarrow \mathsf{PRF}(K, i_j) \\ (\mathsf{vk}_{i_j}, \mathsf{sk}_{i_j}) \leftarrow \mathsf{Sig.KeyGen}_{i_j}(1^\lambda; r_{i_j}) \end{array} \right\}.$$

- WE *be a witness encryption scheme for language $\mathcal{L}'$.*
- SPB *be a SPB hashing scheme;*
- $\mathcal{L}$ *be a language such that*

$$\mathcal{L} = \left\{ \begin{array}{c} (m, \mathsf{ct}, \mathsf{hk}, h, \mathsf{rhk}, rh, x) : \exists(\mathsf{vk}, i, \mathsf{Sig.Verify}, \mathsf{ind}, \tau, \rho, \sigma, r_{\mathsf{ct}}) \ s.t. \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{hk}, h, i, \mathsf{vk}, \tau) \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{rhk}, rh, \mathsf{ind}, \mathsf{Sig.Verify}, \rho) \\ \mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, x, \sigma; r_{\mathsf{ct}}) \\ 1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}, m, \sigma) \end{array} \right\};$$

  where Sig.Verify *is a description of the verification algorithm of a signature scheme* Sig. *We assume again, that for all schemes,* |Sig.Verify| *is bounded by a polynomial $\beta(\lambda)$.*
- NIWI *be a NIWI scheme for the language*

$$\mathcal{L}_{\mathsf{OR}} = \left\{ \begin{array}{c} (m, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1, x) : \\ \exists b \in \{0,1\} \ s.t. \ (m, \mathsf{ct}_b, \mathsf{hk}_b, h_b, \mathsf{rhk}_b, rh_b, x) \in \mathcal{L} \end{array} \right\}.$$

We now describe the scheme in full detail.

$\mathsf{Sign}(1^\lambda, \mathsf{sk}_i, m, R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell), i, S = \{\mathsf{Sig}_i\}_{i\in[M]}\})$:

- *Determine an index* ind *with* $\mathsf{tag}(\mathsf{vk_i}, \mathsf{Sig_{ind}})$. *Parse* $\mathsf{Sig_{ind}} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$. *Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i\in[M]}$ *to be the list of verification algorithms in S.*

- *Compute* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$.
- *Compute* $(\mathsf{hk}_b, \mathsf{shk}_b) \leftarrow \mathsf{SPB.Gen}(1^\lambda, \ell, i)$ *and* $h_b \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_b, R)$ *for* $b \in \{0,1\}$. *Also, compute the proof* $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0, \mathsf{shk}_0, R, i)$.
- *Compute* $(\mathsf{rhk}_b, \mathsf{rshk}_b) \leftarrow \mathsf{SPB.Gen}(1^\lambda, M, \mathsf{ind})$ *and* $rh_b \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_b, S')$ *for* $b \in \{0,1\}$. *Also, compute the proof* $\rho \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_0, \mathsf{rshk}_0, S', \mathsf{ind})$.
- *Encrypt* $\mathsf{ct}_0 \leftarrow \mathsf{WE.Enc}(1^\lambda, x', \sigma; r_{\mathsf{ct}})$ *and* $\mathsf{ct}_1 \leftarrow \mathsf{WE.Enc}(1^\lambda, x', 0)$, *where* $x' = R$.
- *Set* $x = (m, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1, x')$.
- *Set* $w = (\mathsf{vk}, i, \mathsf{Sig.Verify}_{\mathsf{ind}}, \mathsf{ind}, \tau, \rho, \sigma, r_{\mathsf{ct}})$.
- *Compute the proof* $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$.
- *Output* $\Sigma \leftarrow (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$.

$\mathsf{Verify}(\Sigma, m, R, S)$:

- *Parse* $\Sigma = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$.
- *Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i \in [M]}$ *to be the list of verification algorithms in* S.
- *Compute* $h_b \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_b, R)$ *for* $b \in \{0,1\}$.
- *Compute* $rh_b \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_b, S')$ *for* $b \in \{0,1\}$.
- *Set* $x = (m, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1, R)$.
- *If* $1 \leftarrow \mathsf{NIWI.Verify}(x, \pi)$, *output* 1. *Else, output* 0.

Signature Size. A signature for a message m under a ring R (of size ℓ) and a list of schemes S (of size M) is of the form $\Sigma = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$. We first analyze the size of the ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$. The circuit that verifies the relation $\mathcal{R}'$ of language $\mathcal{L}'$ needs to have size at least $\mathcal{O}(\ell \cdot \mathsf{poly}(\lambda))$ since witnesses for this language are of that size. It is clear that the conditions can be checked in a circuit of this size.

Moreover, a similar analysis as the one made for Construction 1 shows that the total size of $(\mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$ is $\mathcal{O}((\log \ell + \log M) \cdot \mathsf{poly}(\lambda))$. We may assume, that $M \leq \ell$ because signature schemes that no corresponding key exists for in R may be omitted without altering functionality.

We conclude that the signatures in this scheme have size $\mathcal{O}(\ell \cdot \mathsf{poly}(\lambda))$.

3.5.2 Proofs

We now show that the construction presented above fulfills the required properties for a URS due to Def. 58. We start by showing correctness. Then we proceed to prove unforgeability and t-anonymity.

Theorem 4 (Correctness)**.** *The scheme presented in Construction 2 is correct, given that* NIWI *is perfectly complete.*

Proof. Let $\lambda \in \mathbb{N}$, $\ell, M = \mathsf{poly}(\lambda)$, and Sig' be a correct signature scheme. Let further $j \in [\ell]$, a messages m and a key pair honestly constructed by $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig}'.\mathsf{KeyGen}(1^\lambda)$ be given. Now, for $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ such that $\mathsf{vk}_j = \mathsf{vk}$ and $S = \{\mathsf{Sig}_i\}_{i \in [M]}$ such that $\mathsf{Sig}' \in S$, we have to show

$$\Pr\left[1 \leftarrow \mathsf{Verify}\left(\mathsf{Sign}(1^\lambda, \mathsf{sk}, m, R, j, S), m, R, S\right)\right] = 1.$$

Clearly, the statement x in $\mathsf{Verify}\left(\mathsf{Sign}(1^\lambda, \mathsf{sk}_j, m, R, j, S), m, R, S\right)$ is identical to the statement in $\mathsf{Sign}(1^\lambda, \mathsf{sk}_j, m, R, j, S)$. Therefore by the perfect completeness of NIWI, it remains to show that $w = (\mathsf{vk}, i, \mathsf{Sig.Verify}_{\mathsf{ind}}, \mathsf{ind}, \tau, \rho, \sigma, r_{\mathsf{ct}})$ was a valid witness. The case $b = 0$ in $\mathcal{L}_{\mathsf{OR}}$ is fulfilled by construction and correctness of Sig' and SPB. Therefore, Vrfy outputs 1. □

Theorem 5 (Unforgeability)**.** *Assume that the challenge signature schemes* $\mathsf{LS} = \{\mathsf{Sig}_i\}_{i\in[M]}$ *are EUF-CMA,* PRF *is a pseudorandom function,* NIWI *is perfectly sound,* SPB *is somewhere perfectly binding and* WE *is correct. Then, the scheme presented in Construction 2 is unforgeable.*

To prove the unforgeability of our URS, we once again guess which underlying signature scheme Sig_{i^*} the adversary will break EUF-CMA unforgeability of. First, we build a hybrid where the experiment computes all verification keys, except for vk_{i^*}, using randomness from a PRF (instead of using truly random coins). Note that this change goes unnoticed given that PRF is a PRF. Next, we build a reduction to the unforgeability of the underlying signature scheme. The idea is similar to the proof of Theorem 2. Namely, the goal of the reduction is to extract a valid signature from either ct_0 or ct_1. To do this, note that the reduction is in possession of the key K such that vk_i is created using random coins $\mathsf{PRF}(K, i)$, for all $i \neq i^*$ where vk_{i^*} is the challenge verification key. Then, by the correctness of the WE and the perfect soundness of the NIWI, the reduction can use K to decrypt both ct_0 and ct_1. In the end, there is a non-negligible probability that the reduction can extract a valid signature under vk_{i^*}, thus breaking the unforgeability of the signature scheme.

Proof. Let $\mathcal{A}$ be a PPT adversary against the unforgeability of URS. Consider the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This is the real unforgeability experiment defined in Def. 60. In particular, all verification keys vk_i computed by the challenger are computed honestly using $\mathsf{Sig.KeyGen}_{\mathsf{ind}_i}$ for indices ind_i provided by the adversary. That is, we compute $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda)$.

Hybrid $\mathcal{H}_1$. This hybrid is identical to the previous one except that the experiment samples $i^* \leftarrow\!\!\$\ [\ell]$. Let $\{i_1, \ldots, i_{\ell-1}\} = [\ell] \setminus \{i^*\}$.

Note that this hybrid is identically distributed from the view of the adversary.

Hybrid $\mathcal{H}_{2,j}$ for $j = \{1, \ldots, \ell - 1\}$. This hybrid is identical to the previous one, except that the challenger sets the verification keys vk_{i_j} to be computed using randomness coming from a PRF. That is, the experiment samples $K \leftarrow\!\!\$\ \{0,1\}^\lambda$ and computes $(\mathsf{vk}_{i_j}, \mathsf{sk}_{i_j}) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_{i_j}}(1^\lambda, r_{i_j})$ where $r_{i_j} \leftarrow \mathsf{PRF}(K, i_j)$.

Claim 10. *Assume that* PRF *is a PRF. Then, hybrids* $\mathcal{H}_1$ *and* $\mathcal{H}_{2,\ell-1}$ *are indistinguishable.*

Proof. We prove that hybrids $\mathcal{H}_{2,j-1}$ and $\mathcal{H}_{2,j}$ are indistinguishable, for all $j \in \{1, \ldots, \ell - 1\}$ and where $\mathcal{H}_{2,0} = \mathcal{H}_1$.

Suppose that there is an adversary $\mathcal{A}$ that is able to distinguish hybrids $\mathcal{H}_{2,j-1}$ and $\mathcal{H}_{2,j}$. Then, there is a reduction $\mathcal{R}$ that breaks the security of the underlying PRF.

The reduction receives ν from the challenger and computes the verification key vk_j by $(\mathsf{vk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_{i_j}}(1^\lambda; \nu)$. For all $i \neq j$, the remaining vk_i are computed as in hybrid $\mathcal{H}_{2,j-1}$. From now on, the reduction behaves exactly as in $\mathcal{H}_{2,j-1}$.

Note that, if $\nu \leftarrow\!\!\$\ \{0,1\}^\beta$ is a uniform string then the simulation is identical to $\mathcal{H}_{2,j-1}$. On the other hand, if $\nu \leftarrow \mathsf{PRF}(K, i_j)$ for some $K \leftarrow\!\!\$\ \{0,1\}^\lambda$ then the simulation is identical to $\mathcal{H}_{2,j}$. Therefore, $\mathcal{R}$ outputs whatever $\mathcal{A}$ outputs and gets the same advantage. □

We now prove that we can reduce the hardness of unforgeability for the scheme in Hybrid $\mathcal{H}_{2,\ell-1}$ to the EUF-CMA of the underlying signature scheme.

Let $\mathcal{A}$ be an adversary that wins in $\mathcal{H}_{2,\ell-1}$. We provide the description of an adversary $\mathcal{B}$ that breaks the EUF-CMA of one of the underlying signature schemes $\mathsf{Sig}_{\mathsf{ind}_{i^*}}$. We will once again guess this index ind_{i^*} first. Then, let $\mathcal{C}$ be the challenger in the EUF-CMA game for $\mathsf{Sig}_{\mathsf{ind}_{i^*}}$ due to Def. 3.

1. $\mathcal{B}$ provides $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i\in[M]}$ to $\mathcal{A}$.
2. Upon receiving the indices $\{\mathsf{ind}_i\}_{i\in[\ell]}$ from $\mathcal{A}$, $\mathcal{B}$ guesses an index $i^* \leftarrow [\ell]$ at random. It samples $K \leftarrow\!\!\$\ \{0,1\}^\lambda$. It creates $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda; r_i)$ for all $i \neq i^*$ where $r_i \leftarrow \mathsf{PRF}(K, i)$ and sets vk_{i^*} to be a verification key under $\mathsf{Sig}_{\mathsf{ind}_{i^*}}$ given by the challenger $\mathcal{C}$ for EUF-CMA security of that scheme. $\mathcal{B}$ sends $R = \{\mathsf{vk}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.
3. $\mathcal{B}$ simulates the oracles $\mathtt{OCorrupt}$, $\mathtt{URSign}$ and $\mathtt{OSign}$ in the following way:
   - $\mathtt{OCorrupt}(i)$: Whenever $\mathcal{A}$ sends a query i, $\mathcal{B}$ reveals sk_i if $i \neq i^*$. Otherwise, it aborts the protocol;
   - $\mathtt{URSign}(m, \bar{R}, i, \bar{S})$: Whenever $\mathcal{A}$ sends a query $(m, \bar{R}, i, \bar{S})$, $\mathcal{B}$ proceeds as in the experiment, for $i \neq i^*$. If $i = i^*$, $\mathcal{B}$ checks whether $\mathsf{vk}_{i^*} \in \bar{R}$ and denotes its position as j^*. If additionally $\mathsf{Sig}_{\mathsf{ind}_{i^*}} \in \bar{S}$, $\mathcal{B}$ sends a query to $\mathcal{C}$ for message m and, upon receiving the signature σ for m, it proceeds to compute a signature Σ as in the hybrid, using σ instead of computing it itself.
   - $\mathtt{OSign}(i, m)$: Whenever $\mathcal{A}$ sends a query (i, m) with $i \neq i^*$, $\mathcal{B}$ sends $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$. Otherwise, it sends a query m to $\mathcal{C}$ and, upon receiving a signature σ, it outputs σ to $\mathcal{A}$.
4. Upon receiving $\Sigma^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \mathsf{rhk}_0^*, \mathsf{rhk}_1^*, \pi^*)$, plus m^*, R^* and S^* from $\mathcal{A}$, $\mathcal{B}$ decrypts $\sigma_0^* \leftarrow \mathsf{WE.Dec}(\mathsf{ct}_0^*, w')$ and $\sigma_1^* \leftarrow \mathsf{WE.Dec}(\mathsf{ct}_1^*, w')$, using the witness $w' = (\{\mathsf{Sig}_{\mathsf{ind}_i}\}_{i\in[\ell]\setminus\{i^*\}}, K)$. Per our construction of the keys, this is a valid witness. If $1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}_{i^*}, m^*, \sigma_0^*)$, it outputs σ_0^*. Else if $1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}_{i^*}, m^*, \sigma_1^*)$, it outputs σ_1^*.

Note that the statement x' is in $\mathcal{L}'$, because $\mathcal{B}$ creates all verification keys vk_i as $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda, r_i)$, where $r_i \leftarrow \mathsf{PRF}(K, i)$ for all $i \neq i^*$. A witness for x' is thus exactly $w' = (\{\mathsf{Sig}_{\mathsf{ind}_i}\}_{i\in[\ell]\setminus\{i^*\}}, K)$.

Following a similar analysis as the one of Theorem 2, we conclude that $\mathcal{B}$ succeeds in extracting a valid forgery with probability at least $\frac{1}{\ell^2}\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Unf}}\right]$.

Remark also that $\mathcal{B}$ only takes polynomial time to output a valid forgery, given that $\mathcal{A}$ runs in polynomial time. This concludes the proof of the theorem. □

Theorem 6 (t-Anonymity)**.** *Assume that* NIWI *is witness-indistinguishable,* SPB *is index hiding and* WE *is soundness secure. Then the scheme presented in Construction 2 is t-anonymous where $t = (\lambda - \omega(\log\lambda))/q$ and q is a lower bound of the min-entropy of verification keys in the ring.*

The proof of the theorem is similar to the proof of Theorem 3. However, now we would like to use the security of the WE to replace ct_1 by an encryption of a valid signature under one key (and then replace back by an encryption of 0). To do this, we note that (unlike the unforgeability security proof described above) all verification keys in R are computed using truly random coins. The challenge ring given by the adversary must include at least t of these keys. A simple information-theoretical argument states that there is only a negligible probability that there is a PRF key K such that $t-1$ of these honestly generated verification keys are malformed. This is because they are sampled independently and thus it is unlikely that they are correlated

via a PRF key. Hence, we can conclude that $\ell - 1$ verification keys in the adversary's ring are not created using random coins $\mathsf{PRF}(K, i)$, except with negligible probability. In other words, there is a negligible probability that $x' \in \mathcal{L}'$. We can thus use the security of the WE to safely replace encryptions of signatures and encryptions of 0. That is, we switch out the encrypted signature in ct_0 from one under one challenge key to a signature under another one.

Proof. Let $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i\in[M]}$ be a list of unforgeable signature schemes. In the following we will only modify our response to $\mathcal{A}$ after they made their challenge query. That means, we have already received indices $\{\mathsf{ind}_i\}_{i\in\ell}$ from $\mathcal{A}$ and provided them with a ring R of verification keys and respective random coins. Let now $(m^*, R^* = (\mathsf{vk}'_1, \ldots, \mathsf{vk}'_p), S^* = (\mathsf{Sig}'_1, \ldots, \mathsf{Sig}'_q), (j_k)_{k\in[t]})$ be the challenge query of $\mathcal{A}$ in the game of Def. 61. Let $j_0 = j_k$ for any $k \in \{2, 3, \ldots, t\}$. We show that the encryptions using vk'_{j_0} and vk'_{j_1} are not distinguishable. Let i_0, i_1 be the indices for these challenge keys in K, that is $\mathsf{vk}_{i_0} = \mathsf{vk}'_{j_0}, \mathsf{vk}_{i_1} = \mathsf{vk}'_{j_1}$ in R^* and by $\mathsf{ind}'_0, \mathsf{ind}'_1$ the indices of their corresponding signature verification schemes in S^*. The proof of the theorem follows from the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This is the real anonymity experiment defined in Def. 61 where the key at index j_0 is used in encryption. That is, the challenger outputs a signature $\Sigma^* = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$ where

$$\begin{aligned}
\sigma_0 &\leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_{i_0}}(\mathsf{sk}_{i_0}, m^*)\\
\mathsf{ct}_0 &\leftarrow \mathsf{WE.Enc}(1^\lambda, x', \sigma_0; r_{\mathsf{ct}})\\
\mathsf{ct}_1 &\leftarrow \mathsf{WE.Enc}(1^\lambda, x', 0)\\
\pi &\leftarrow \mathsf{NIWI.Prove}(x, w_0)
\end{aligned}$$

where $x = (m, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1, R^*)$, $x' = R^*$ and $w_0 = (\mathsf{vk}, j_0, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_0}}, \mathsf{ind}'_0, \tau, \rho, \sigma_0, r_{\mathsf{ct}})$.

Hybrid $\mathcal{H}_1$. This hybrid is identical to the previous one, except that the challenger sets $(\mathsf{hk}_1, \mathsf{shk}_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |R^*|, j_1)$. Additionally, it computes $\tau' \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_1, \mathsf{shk}_1, R^*, j_1)$. Also, it sets $(\mathsf{rhk}_1, \mathsf{rshk}_1) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |S'|, \mathsf{ind}'_1)$. Additionally, it computes $\rho' \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_1, \mathsf{rshk}_1, S', \mathsf{ind}'_1)$.

Claim 11. *Assume that* SPB *is index hiding. Then hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 1.

Hybrid $\mathcal{H}_2$. This hybrid is identical to the previous, except that the challenger computes $\sigma_1 \leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_{i_1}}(\mathsf{sk}_{i_1}, m^*)$ and sets $\mathsf{ct}_1 \leftarrow \mathsf{WE.Enc}(1^\lambda, x', \sigma_1; r'_{\mathsf{ct}})$.

Claim 12. *Assume that* WE *is soundness secure. Then hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$ are indistinguishable.*

Proof. First, note that all verification keys vk_i given to the adversary $\mathcal{A}$ are generated as $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda; r_i)$ where $r_i \leftarrow\!\!\$\ \{0,1\}^\lambda$ (here we explicitly input random coins r_i into the key generation algorithm).

We claim that, if each $r_i \leftarrow\!\!\$\ \{0,1\}^\lambda$, then $x' \notin \mathcal{L}'$ except with negligible probability. To prove this, we show that if $\mathsf{vk}_i \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, r_i)$ for $r_i \leftarrow\!\!\$\ \{0,1\}^\lambda$ then there is a negligible probability that there exists a PRF key K such that $\mathsf{vk}'_i \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, \mathsf{PRF}(K, i))$ and $\mathsf{vk}_i = \mathsf{vk}'_i$ for at least $t - 1$ keys.

Assume that the min-entropy of each verification key is at least q bits, that is, $\mathbb{H}_\infty(\mathsf{vk}_i) \geq q$. Since each vk_i is independently sampled then $\mathbb{H}_\infty(\mathsf{vk}_1, \ldots, \mathsf{vk}_{t-1}) \geq (t-1)q$. On the other hand, the number of $t-1$ tuples $(\mathsf{vk}'_1, \ldots, \mathsf{vk}'_{t-1})$ such that each vk'_i is created using $\mathsf{vk}'_i \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, \mathsf{PRF}(K, i))$ is 2^λ. This is because of two reasons: i) once we choose the PRF key $K \in \{0,1\}^\lambda$, the verification keys vk'_i are fixed; and ii) the number of different PRF keys is 2^λ.

We now compute the probability that $(\mathsf{vk}_1, \ldots, \mathsf{vk}_{t-1}) = (\mathsf{vk}'_1, \ldots, \mathsf{vk}'_{t-1})$ where $\mathsf{vk}'_i \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, \mathsf{PRF}(K, i))$ for some $K \in \{0,1\}^\lambda$.
Since $\mathbb{H}_\infty(\mathsf{vk}_1, \ldots, \mathsf{vk}_{t-1}) \geq (t-1)q$, by definition of min-entropy it holds that for any fixed tuple $(\alpha_1, \ldots, \alpha_{t-1})$ in the range of $\mathsf{Sig.KeyGen}$ we have that the probability of all vk_i cooinciding with the α_i is bounded by $\Pr[(\mathsf{vk}_1, \ldots, \mathsf{vk}_{t-1}) = (\alpha_1, \ldots, \alpha_{t-1})] \leq 2^{-(t-1)q}$. Hence,

$$\Pr\left[\begin{array}{c}(\mathsf{vk}_1, \ldots, \mathsf{vk}_{t-1}) = \\ (\mathsf{vk}'_1, \ldots, \mathsf{vk}'_{t-1})\end{array} : \begin{array}{c}\mathsf{vk}'_i \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, \mathsf{PRF}(K, i)) \\ \text{for some } K \in \{0,1\}^\lambda\end{array}\right] \leq 2^\lambda \frac{1}{2^{(t-1)q}} = 2^{\lambda-(t-1)q}$$

where the first inequality is obtained by applying a union bound over the number of possible different PRF keys. Setting $t \geq (\lambda + \omega(\log \lambda))/q + 1$, we get that the probability above is negligible in λ. Hence $x' \notin \mathcal{L}$ except with negligible probability.

Now, since $x' \notin \mathcal{L}'$ and given an adversary that distinguishes both games, we can easily build a reduction $\mathcal{R}$ that breaks the soundness security of the underlying WE. The reduction simply sets $m_0 = 0$ and $m_1 = \sigma_1$ as the messages to be sent to the challenger of the soundness security of WE. Upon receiving a ciphertext ct from the challenger, it sets $\mathsf{ct}_1 = \mathsf{ct}$ in an otherwise perfect simulation of $\mathcal{H}_0$. If $b = 0$ (in the soundness security game), then the game played by $\mathcal{A}$ is identical to $\mathcal{H}_0$, otherwise it is identical to $\mathcal{H}_1$. Therefore, $\mathcal{R}$ outputs whatever $\mathcal{A}$ outputs and gets the same advantage. □

Hybrid $\mathcal{H}_3$. This hybrid is identical to the previous one, except that the challenger computes $\pi \leftarrow \mathsf{NIWI.Prove}(x, w_1)$ where $w_1 = (\mathsf{vk}_{i_1}, j_1, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_1}}, \mathsf{ind}'_1, \tau', \rho', \sigma_1, r'_{\mathsf{ct}})$.

Claim 13. *Assume that* NIWI *is witness indistinguishable. Then hybrids* $\mathcal{H}_2$ *and* $\mathcal{H}_3$ *are indistinguishable.*

The proof of the claim is identical to the proof of Claim 4.

Hybrid $\mathcal{H}_4$. This hybrid is identical to the previous, except that the challenger computes $\sigma'_1 \leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_{i_1}}(\mathsf{sk}_{i_1}, m^*)$ and sets $\mathsf{ct}_0 \leftarrow \mathsf{WE.Enc}(1^\lambda, x', \sigma'_1; r''_{\mathsf{ct}})$.

Claim 14. *Assume that* WE *is soundness secure. Then hybrids* $\mathcal{H}_3$ *and* $\mathcal{H}_4$ *are indistinguishable.*

The proof of the claim is identical to the proof of Claim 12.

Hybrid $\mathcal{H}_5$. This hybrid is identical to the previous one, except that the challenger sets $(\mathsf{hk}_0, \mathsf{shk}_0) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |R^*|, j_1)$. It computes $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0, \mathsf{shk}_0, R^*, j_1)$. Also, it computes $(\mathsf{rhk}_0, \mathsf{rshk}_0) \leftarrow \mathsf{SPB.Gen}(1^\lambda, |S'|, \mathsf{ind}'_1)$ and $\rho \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_0, \mathsf{rshk}_0, S', \mathsf{ind}'_1)$.

Claim 15. *Assume that* SPB *is index hiding. Then hybrids* $\mathcal{H}_4$ *and* $\mathcal{H}_5$ *are indistinguishable.*

The proof of the claim is identical to the proof of Claim 1.

Hybrid $\mathcal{H}_6$. This hybrid is identical to the previous one except that the challenger sets $\pi \leftarrow \mathsf{NIWI.Prove}(x, w_2)$ for $w_2 = (\mathsf{vk}_{i_1}, j_1, \mathsf{Sig.Vrfy}_{\mathsf{ind}_{i_1}}, \mathsf{ind}'_1, \tau, \rho, \sigma'_1, r''_{\mathsf{ct}})$.

Claim 16. *Assume that* NIWI *is witness indistinguishable. Then hybrids $\mathcal{H}_5$ and $\mathcal{H}_6$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 4.

Hybrid $\mathcal{H}_7$. This hybrid is identical to the previous one except that the challenger sets $\mathsf{ct}_1 \leftarrow \mathsf{WE.Enc}(1^\lambda, x', 0)$.

Claim 17. *Assume that* WE *is soundness secure. Then hybrids $\mathcal{H}_6$ and $\mathcal{H}_7$ are indistinguishable.*

The proof of the claim is identical to the proof of Claim 12.

This hybrid is identical to the real anonymity experiment of Def. 61 where the key at index j_1 is used in encryption. This concludes the proof of the theorem. □

3.6 Compact Witness Encryption for Threshold Conjunction Languages

In this section we present a WE scheme that is compact for threshold conjunction languages. We first define the notion of threshold conjunction languages.

Definition 62 (Threshold Conjunction Languages)**.** *Let $\mathcal{L}$ be an* NP *language, with relation $\mathcal{R}$. We define a (t, N)-threshold conjunction language $\mathcal{L}'$ as follows*

$$\mathcal{L}' = \left\{(x_1, \ldots, x_N) : \exists \{i_j\}_{j\in[t]} \in [N] \text{ s.t. } x_{i_j} \in \mathcal{L}\right\}.$$

In other words, an accepting instance $(x_1, \ldots, x_N)$ of $\mathcal{L}'$ is one such that there are at least t accepting instances x_{i_j}.

3.6.1 Construction from Indistinguishability Obfuscation

We now describe our WE scheme for any (t, N)-threshold conjunction language $\mathcal{L}'$. The protocol achieves compact ciphertexts, i.e., of size $\mathcal{O}(\log N)$, when $N - t \in \mathcal{O}(\log N)$.

Construction 3. *Let $N \in \mathsf{poly}(\lambda)$ and t be such that $N - t \in \mathcal{O}(\log N)$ and $\mathcal{L}$ be an* NP *language. Let*

- LSS *be a (t, N)-LSS scheme. In the following, we assume that shares can be written as strings in $\{0,1\}^\lambda$.*
- WE *be a (non-compact) WE scheme for language $\mathcal{L}$.*
- iO *be an obfuscator for all circuits.*
- PPRF *be a puncturable PRF.*
- SSB *be an SSB hashing scheme.*

Additionally, consider the following circuit $\mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N]$ which has the values λ, hk, h, k_0, k_1, t and N hardwired.

$\mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N](i, \tau_i, x_i)$:

- *If* $0 \leftarrow \mathsf{SSB.Verify}(\mathsf{hk}, h, i, x_i, \tau_i)$ *or* $i \geq t$*, return* $\perp$.
- *Compute* $s_i \leftarrow \mathsf{PPRF.Eval}(k_0, i)$ *and random coins* $r_i \leftarrow \mathsf{PPRF.Eval}(k_1, i)$.
- *Compute* $\mathsf{ct}_i \leftarrow \mathsf{WE.Enc}(1^\lambda, x_i, s_i; r_i)$. *Output* ct_i.

We now define the WE scheme for the (t, N)-conjunction language $\mathcal{L}'$.

$\mathsf{Enc}(1^\lambda, x, m)$:

- *Parse* $x = (x_1, \ldots, x_N)$.
- *Create PPRF keys* $k_0 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$ *and* $k_1 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$.
- *For* $i \in [t-1]$, *compute pseudorandom shares* $s_i \leftarrow \mathsf{PPRF}(k_0, i)$. *Compute the remaining shares* $(s_t, \ldots, s_N) \leftarrow \mathsf{LSS.RemainShare}(m, s_1, \ldots, s_{t-1})$.
- *Compute* $\mathsf{hk} \leftarrow \mathsf{SSB.Gen}(1^\lambda, t-1, j)$ *for* $j \leftarrow\!\!\$\ [t-1]$. *Moreover, compute* $h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, \{x_1, \ldots, x_{t-1}\})$.
- *Consider the circuit* $\mathcal{C} = \mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N]$. *Compute* $\bar{\mathcal{C}} \leftarrow \mathsf{iO}(1^\lambda, \mathcal{C})$.
- *For* $i \in \{t, \ldots, N\}$, *compute encryptions* $\mathsf{ct}_i \leftarrow \mathsf{WE.Enc}(1^\lambda, x_i, s_i)$.
- *Output* $\mathsf{ct} = (\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}, \bar{\mathcal{C}}, \mathsf{hk})$.

$\mathsf{Dec}(\mathsf{ct}, w)$:

- *Parse* ct *as* $(\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}, \bar{\mathcal{C}}, \mathsf{hk})$ *and* $w = (w_{i_1}, \ldots, w_{i_t})$.
- *For* $i \in [t-1]$, *compute* $\tau_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, \{x_1, \ldots, x_{t-1}\}, i)$ *and run* $\mathsf{ct}_i \leftarrow \bar{\mathcal{C}}(i, \tau_i, x_i)$.
- *For* $j \in [t]$, *decrypt* $s_{i_j} \leftarrow \mathsf{WE.Dec}(\mathsf{ct}_{i_j}, w_{i_j})$.
- *Reconstruct* $m \leftarrow \mathsf{LSS.Reconstruct}(s_{i_1}, \ldots, s_{i_t})$. *Output* m.

Ciphertext Size. The ciphertext is of the form $(\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}, \bar{\mathcal{C}}, \mathsf{hk})$. Assume that the language $\mathcal{L}$ has a verification circuit $\mathcal{C}_\mathcal{L}$ and $N - t \in \mathcal{O}(\log(N))$.

The ciphertexts ct_i for $i \in \{t, \ldots, N\}$ have size $\mathcal{O}(|\mathcal{C}_\mathcal{L}| \cdot \mathsf{poly}(\lambda))$. Since $N - t \in \mathcal{O}(\log(N))$, then the size of $\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}$ is $\mathcal{O}(\log(N) \cdot |\mathcal{C}_\mathcal{L}| \cdot \mathsf{poly}(\lambda))$.

The obfuscated circuit $\mathcal{C}$ implements the $\mathsf{SSB.Verify}$ algorithm which is of size $\mathcal{O}(\log(N))$. Moreover, all other operations in $\mathcal{C}$ are independent of N and depend only on $|\mathcal{C}_\mathcal{L}|$. Hence, $|\mathcal{C}| \in \mathcal{O}(\log(N) \cdot |\mathcal{C}_\mathcal{L}| \cdot \mathsf{poly}(\lambda))$.

Finally, the hashing key hk is of size $\mathcal{O}(\log(N))$ by the efficiency requirements of SSB.

We conclude that the scheme presented above outputs ciphertexts of size $\mathcal{O}(\log(N) \cdot |\mathcal{C}_\mathcal{L}| \cdot \mathsf{poly}(\lambda))$.

3.6.2 Proofs

We now prove that the witness encryption scheme described above is correct and soundness secure following our definition of witness encryption, Def. 48.

Theorem 7 (Correctness)**.** *The scheme presented in Construction 3 is correct, given that* LSS, SSB *and* WE *are correct.*

Proof. Assume that $x = (x_1, \ldots, x_N) \in \mathcal{L}'$. That is, there exists indices $i_1, \ldots, i_t$ such that $x_{i_j} \in \mathcal{L}$. Let $w_{i_1}, \ldots, w_{i_t}$ be the corresponding witnesses.

First, note that for all $i \in [t]$ and by the correctness of the SSB hashing scheme, the circuit $\mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N](i, \tau_i, x_i)$ always outputs $\mathsf{ct}_i \leftarrow \mathsf{WE.Enc}(1^\lambda, x_i, s_i)$ since $1 \leftarrow \mathsf{SSB.Verify}(\mathsf{hk}, h, i, x_i, \tau_i)$ for $\mathsf{hk} \leftarrow \mathsf{SSB.Gen}(1^\lambda, t-1, j)$ (for $j \leftarrow\!\!\$\ [t-1]$), $h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, \{x_1, \ldots, x_{t-1}\}$ and $\tau_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, \{x_1, \ldots, x_{t-1}\}, i)$.

Moreover, $s_{i_j} \leftarrow \mathsf{WE.Dec}(\mathsf{ct}_{i_j}, w_{i_j})$ holds by the correctness of the WE scheme, giving us access to t of N shares. Finally, the correctness of the LSS scheme implies that we succesfully extract $m \leftarrow \mathsf{LSS.Reconstruct}(s_{i_1}, \ldots s_{i_t})$. □

Theorem 8 (Soundness security)**.** *The scheme presented in Construction 3 is soundness secure given that* SSB *is index hiding and somewhere statistically binding,* iO *is a secure iO obfuscator,* PPRF *is pseudorandom at punctured points,* WE *is soundness secure and* LSS *is private.*

Before presenting the formal proof, we give a brief outline of it. The proof follows a sequence of hybrids, where the last one can be reduced to the privacy of the LSS. First, note that if $x \notin \mathcal{L}'$, then there do not exist t instances $x_i \in \mathcal{L}$. Assume, for simplicity that $t = N$, then there exists an index i^* such that $x_{i^*} \notin \mathcal{L}$. We start with a hybrid that is identical to the real soundness security game.

Then, we use the index hiding of the SSB hashing scheme to replace hk by a hashing key that is binding to index i^*. We then use the *puncturing* technique of Sahai and Waters [SW14]. That is, we create punctured PRF keys k'_0 and k'_1 (by puncturing the PPRF keys k_0 and k_1 respectively) at the point i^*. At the same time, we embed into the obfuscated circuit the ciphertext $\mathsf{ct}_{i^*} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i^*}, s_{i^*}; r_{i^*})$ where $s_{i^*} \leftarrow \mathsf{PPRF.Eval}(k_0, i^*)$ and $r_{i^*} \leftarrow \mathsf{PPRF.Eval}(k_1, i^*)$. Given that the SSB key is somewhere statistically binding at the point i^*, the circuits are functionally equivalent and we can use the security of the $i\mathcal{O}$ obfuscator to argue indistinguishability. We can now replace the values s_{i^*}, r_{i^*} by uniform ones since the PPRF is pseudorandom at punctured points. Finally, we replace ct_{i^*} by an encryption of 0. To conclude the proof, we can easily build a reduction to the security of the LSS.

In the more general case, some WE encryptions with respect to false statements are computed using the obfuscated program and some of them are given in the plain. For the former ones, we simply repeat the process above. For the latter ones, we use the security of the WE to replace these encryptions by encryptions of 0.

Proof. Assume that $x \notin \mathcal{L}'$. That is, there exists $x_{i_j} \notin \mathcal{L}$ for $j \in [N - t + 1]$ (where $N - t + 1 \in \log(N)$). Let $\delta \in [N]$ be such that $i_1, \ldots, i_\delta \leq t - 1 < i_{\delta+1}, \ldots, i_{N-t+1}$. The proof of the theorem follows from the following sequence of hybrids:

$$\begin{aligned}
\mathcal{H}_0 &\approx_c \mathcal{H}_{1,1} \approx_c \cdots \approx_c \mathcal{H}_{1,5} \\
&\approx_c \mathcal{H}_{2,1} \approx_c \cdots \approx_c \mathcal{H}_{2,5} \\
&\vdots \\
&\approx_c \mathcal{H}_{\delta,1} \approx_c \cdots \approx_c \mathcal{H}_{\delta,5} \\
&\approx_c \mathcal{H}_{\delta+1,1} \approx_c \cdots \approx_c \mathcal{H}_{\delta+1,N-t+1-\delta}
\end{aligned}$$

The first hybrid $\mathcal{H}_0$ denotes the real soundness security experiment of Def. 50. The last hybrid can be reduced to the privacy of the underlying LSS.

Hybrid $\mathcal{H}_0$. This is the real soundness security game as defined in Def. 50.

Hybrid $\mathcal{H}_{j,1}$ for $j = 1, \ldots, \delta$. This hybrid is identical to the previous one, if $j = 1$, and identical to $\mathcal{H}_{j-1,5}$ otherwise, except that the challenger sets $\mathsf{hk} \leftarrow \mathsf{SSB.Gen}(1^\lambda, t - 1, i_j)$ if $i_j \leq t - 1$.

Claim 18. *Assume that* SSB *is index hiding. Then hybrids* $\mathcal{H}_{j-1,0}$ *and* $\mathcal{H}_{j,1}$ *are indistinguishable, for* $j \in \{1, \ldots, \delta\}$ *where* $\mathcal{H}_{0,0} = \mathcal{H}_0$ *and* $\mathcal{H}_{j-1,0} = \mathcal{H}_{j-1,5}$ *(defined below).*

The proof of the claim is similar to the proof of Claim 1.

We introduce an event $\mathbf{bad}_j$ for $j \in \{1, \ldots, \delta\}$, which is fulfilled if the fresh honestly generated key $\mathsf{hk} \leftarrow \mathsf{SSB.Gen}(1^\lambda, t-1, i_j)$ in $\mathcal{H}_{j,1}$ is *not* statistically binding.

Hybrid $\mathcal{H}_{j,2}$ for $j = 1, \ldots, \delta$. This hybrid is identical to the previous one except that, if $i_j \leq t - 1$, the challenger computes the keys $k'_0 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$ and $k'_1 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$, the punctured keys $k_{i_j} \leftarrow \mathsf{PPRF.Puncture}(k'_0, S)$ and $k'_{i_j} \leftarrow \mathsf{PPRF.Puncture}(k'_1, S)$ where $S = \{i_1, \ldots, i_j\}$, and sets $k_0 = k_{i_j}$ and $k_1 = k'_{i_j}$.

Moreover, it computes $\mathsf{ct}_{i_j} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i_j}, s_{i_j}; r_{i_j})$ where $s_{i_j} \leftarrow \mathsf{PPRF.Eval}(k'_0, i_j)$ and $r_{i_j} \leftarrow \mathsf{PPRF.Eval}(k'_1, i_j)$.

Additionally, the challenger modifies

$$\mathcal{C} = \mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N, \mathsf{ct}_{i_1}, \ldots, \mathsf{ct}_{i_{j-1}}]$$

to a circuit

$$\mathcal{D} = \mathcal{D}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N, \mathsf{ct}_{i_1}, \ldots, \mathsf{ct}_{i_{j-1}}, \mathsf{ct}_{i_j}]$$

that behaves exactly as $\mathcal{C}$ except on input (i, τ_i, x_i) with $i = i_j$. In this case, the circuit $\mathcal{D}$ first checks if $1 \leftarrow \mathsf{SSB.Verify}(\mathsf{hk}, h, i, x_i, \tau_i)$ and, if so, it outputs ct_{i_j} that is hardwired.

This hybrid is defined for $j = 1, \ldots, \delta$. Note that the number of hardcoded ciphertexts is bounded by $\delta \leq N - t + 1 = \mathcal{O}(\log(N))$.

Claim 19. *Assume that* iO *is a secure iO obfuscator for all circuits and* SSB *is somewhere statistically binding. Then hybrids* $\mathcal{H}_{j,1}$ *and* $\mathcal{H}_{j,2}$ *are indistinguishable for* $j = \{1, \ldots, \delta\}$.

Proof. Given an adversary $\mathcal{A}'$ that is able to distinguish both hybrids, we can build a reduction $\mathcal{R}$ that breaks the security of the underlying iO scheme.

To see this, first observe that the circuits $\mathcal{C}$ and $\mathcal{D}$ are functionally equivalent. That is, for every input (i, τ_i, x_i), we claim that

$$\mathcal{C}(i, \tau_i, x_i) = \mathcal{D}(i, \tau_i, x_i).$$

This fact can be established by noting that the only possible inputs where $\mathcal{C}$ and $\mathcal{D}$ could differ are of the form $(i_j, \cdot, \cdot)$.

Let us consider two cases: Case one is that $\mathbf{bad}_j$ happened. By SSB being statistically binding this happens with negligible probability. We accept that the simulation fails in this negligible case.
In the other case the SSB key hk is binding at position i_j and we have that there exists only one possible form of inputs $(i_j, \cdot, \cdot)$ that output something different then $\perp$. For both circuits that is an input $(i_j, \tau_{i_j}, x_{i_j})$ for the correct x_{i_j} for index i_j and a verifying SSB proof τ_{i_j}.

For this type of input $\mathcal{D}$ outputs the hardwired ciphertext $\mathsf{ct}_{i_j} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i_j}, s_{i_j}; r_{i_j})$, where $s_{i_j} \leftarrow \mathsf{PPRF.Eval}(k'_0, i_j)$ and $r_{i_j} \leftarrow \mathsf{PPRF.Eval}(k'_1, i_j)$. This coincides with the output of $\mathcal{C}$ by definition.

We now describe the reduction $\mathcal{R}$ against security of iO. The reduction $\mathcal{R}$ chooses

$$\mathcal{C} = \mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N, \mathsf{ct}_{i_1}, \ldots, \mathsf{ct}_{i_{j-1}}]$$

and

$$\mathcal{D} = \mathcal{D}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N, \mathsf{ct}_{i_1}, \ldots, \mathsf{ct}_{i_{j-1}}, \mathsf{ct}_{i_j}]$$

as challenge circuits. Upon receiving the obfuscated circuit $\bar{\mathcal{C}}$ from the challenger, $\mathcal{R}$ simply outputs the ciphertext $\mathsf{ct} = (\{\mathsf{ct}_i\}_{i \in \{t, \ldots, N\}}, \bar{\mathcal{C}}, \mathsf{hk})$ (where all other values are computed as in hybrid $\mathcal{H}_{j,1}$).

Remark that, if $\bar{\mathcal{C}} \leftarrow \mathsf{iO}(1^\lambda, \mathcal{C})$, then the game is identical to $\mathcal{H}_{j,1}$. Else if $\bar{\mathcal{C}} \leftarrow \mathsf{iO}(1^\lambda, \mathcal{D})$ the game is identical to $\mathcal{H}_{j,2}$. The reduction outputs the same as $\mathcal{A}$ and has the same advantage minus the negligible advantage of $\mathbf{bad}_j$ happening. □

Hybrid $\mathcal{H}_{j,3}$ for $j = 1, \ldots, \delta$. This hybrid is identical to the previous one except that, if $i_j \leq t - 1$, the challenger samples $r_{i_j} \leftarrow\!\!\$\ \{0,1\}^\lambda$.

Remark on Ciphertext Size. Note that we are puncturing the PPRF on at most $N-t+1 \in \mathcal{O}(\log N)$ points. So the size of the punctured key is $\mathcal{O}(\log N \cdot \mathsf{poly}(\lambda))$. This means the size of the ciphertext of our WE scheme does not exceed $\mathcal{O}(\log N \cdot \mathsf{poly}(\lambda))$.

Claim 20. *Assume that* PPRF *is pseudorandom at punctured points. Then hybrids* $\mathcal{H}_{j,2}$ *and* $\mathcal{H}_{j,3}$ *are indistinguishable.*

Proof. Let $\mathcal{A}'$ be an adversary that distinguishes both hybrids. We build a reduction $\mathcal{R}$ that breaks the pseudorandomness at punctured points of the underlying PPRF.

The reduction works as follows: It sends $S = \{i_1, \ldots, i_j\}$ to the PPRF challenger. Upon receiving k_S and the challenge $T = \{y_1, \ldots, y_j\}$, the reduction behaves exactly as in hybrid $\mathcal{H}_{j,2}$ except that it sets $k_1 = k_S$ and $r_{i_j} = y_j$.

Observe that, if $y_j \leftarrow \mathsf{PPRF.Eval}(k, i_j)$ then the simulated game is identical to $\mathcal{H}_{j,2}$. Else if y is uniformly chosen, then the simulated game is identical to $\mathcal{H}_{j,3}$. We output whatever $\mathcal{A}$ outputs and conclude that $\mathcal{R}$ breaks the pseudorandomness at punctured points with exactly the same advantage that $\mathcal{A}$ has in distinguishing the hybrids. □

Hybrid $\mathcal{H}_{j,4}$ for $j = 1, \ldots, \delta$. This hybrid is identical to the previous one except that, if $i_j \leq t-1$, the challenger samples $s_{i_j} \leftarrow\!\!\$\ \{0,1\}^\lambda$.

Claim 21. *Assume that* PPRF *is pseudorandom at punctured points. Then hybrids* $\mathcal{H}_{j,3}$ *and* $\mathcal{H}_{j,4}$ *are indistinguishable.*

The claim follows by a similar argument as the one of Claim 20.

Hybrid $\mathcal{H}_{j,5}$ for $j = 1, \ldots, \delta$. This hybrid is identical to the previous one except that, for the next index $i_j \leq t-1$ with $x_{i_j} \notin \mathcal{L}$, the challenger computes ct_{i_j} as $\mathsf{ct}_{i_j} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i_j}, 0)$. This is one of the hardwired ciphertexts in the circuit.

Claim 22. *Assume that* WE *is soundness secure. Then hybrids* $\mathcal{H}_{j,4}$ *and* $\mathcal{H}_{j,5}$ *are indistinguishable.*

Proof. Let $\mathcal{A}$ be an adversary that is able to distinguish the hybrids. We build a reduction $\mathcal{R}$ that breaks the soundness security of the underlying WE scheme. We note that by definition $x_{i_j} \notin \mathcal{L}$.

The reduction simply sends the challenge messages $m_0 = s_{i_j}$ and $m_1 = 0$. Upon receiving ct from the challenge, $\mathcal{R}$ behaves exactly as in hybrid $\mathcal{H}_{j,4}$ except that it sets $\mathsf{ct}_{i_j} = \mathsf{ct}$. Note that if $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i_j}, s_{i_j})$ then the game is identical to $\mathcal{H}_{j,4}$, whereas if $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i_j}, 0)$ then the game is identical to $\mathcal{H}_{j,5}$. We output whatever $\mathcal{A}$ outputs and conclude that $\mathcal{R}$ breaks the soundness security with exactly the same advantage that $\mathcal{A}$ has in distinguishing the hybrids. □

Hybrid $\mathcal{H}_{\delta+1,j}$ for $j = 1, \ldots, N-t+1-\delta$. This hybrid is identical to $\mathcal{H}_{\delta,5}$ for $j = 1$ and identical to $\mathcal{H}_{\delta+1,j-1}$ otherwise, except that for the next index $i_{\delta+j}$ for which we have $t-1 < i_{\delta+j} \leq N$ and $x_{i_{j+\delta}} \notin \mathcal{L}$ by definition, the challenger computes $\mathsf{ct}_{i_{j+\delta}}$ as $\mathsf{ct}_{i_{j+\delta}} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i_{j+\delta}}, 0)$.

Claim 23. *Assume that* WE *is soundness secure. Then hybrids* $\mathcal{H}_{\delta+1,j-1}$ *and* $\mathcal{H}_{\delta+1,j}$ *are indistinguishable, for* $j = 1, \ldots, N-t+1-\delta$ *where* $\mathcal{H}_{\delta+1,0} = \mathcal{H}_{\delta,5}$.

The claim follows by a similar argument as the one of Claim 22.

We finally show that the advantage of the adversary is negligible given that the LSS is private.

Claim 24. *Assume that* LSS *is private. Then the advantage of* $\mathcal{A}$ *in* $\mathcal{H}_{\delta+1,N-t+1-\delta}$ *is negligible.*

Proof. Let $\mathcal{A}$ be an adversary that breaks the soundness security of our scheme in hybrid $\mathcal{H}_{\delta+1,N-t+1-\delta}$. We show that there is a reduction $\mathcal{R}$ that uses $\mathcal{A}$ and breaks the privacy of LSS. $\mathcal{A}$ starts by sending (m_0, m_1), which $\mathcal{R}$ redirects to the LSS challenger. $\mathcal{R}$ receives $(\bar{s}_{i_1^*}, \ldots, \bar{s}_{i_{t-1}^*})$. It sets $s_{i_j^*} = \bar{s}_{i_j^*}$ for all $i_j^* \in [N] \setminus \{i_1, \ldots, i_{N-t+1}\}$. Then, it behaves exactly as in hybrid $\mathcal{H}_{\delta+1,N-t+1-\delta}$. Upon receiving a bit b'' from $\mathcal{A}$, $\mathcal{R}$ outputs b''.

Note that the game is identical to the game in $\mathcal{H}_{\delta+1,N-t+1-\delta}$ with challenge bit $b' = b$ where b is the bit of the LSS game. Hence, $\mathcal{R}$ has the same advantage as $\mathcal{A}$. □

This concludes the proof of the theorem. □

3.6.3 Compact Universal Ring Signature from Compact WE for Threshold Conjunction Languages

Consider again the URS construction of § 3.5. One of the requirements of this URS scheme is a (non-compact) WE for a language $\mathcal{L}'$ which is itself a $(N-1, N)$-threshold conjunction language. When we plug the WE scheme for (t, N)-threshold conjunction languages as a drop in replacement for non-compact WE, we obtain a compact URS scheme.

Specifically, the following theorem is a direct consequence of plugging the compact WE scheme for (t, N)-threshold conjunction languages described above with the URS signature from § 3.5.

Theorem 9. *Let*

- $\mathsf{PRG} : \{0,1\}^{\lambda/2} \to \{0,1\}^{\lambda}$ *be a PRG.*
- $\mathcal{L}'$ *be the* $(\ell - 1, \ell)$ *threshold conjunction language defined in Construction 2.*
- WE *be a compact witness encryption scheme for the* $(\ell-1, \ell)$ *threshold conjunction language* $\mathcal{L}'$*. As we have just established, this primitive can be built from secure* $i\mathcal{O}$*,* $(\ell - 1, \ell)$*-LSS, (non-compact) WE for* NP*, PPRF and SSB.*
- SPB *be a SPB hashing scheme;*
- $\mathcal{L}$ *and* $\mathcal{L}_{\mathsf{OR}}$ *be the languages defined in Construction 2.*
- NIWI *be a NIWI scheme for* $\mathcal{L}_{\mathsf{OR}}$*.*

Then there exists a URS scheme that satisfies correctness, anonymity and unforgeability. Moreover, a signature Σ *with respect to a ring of users* R *and a ring of signature schemes* S *has size* $|\Sigma| \in \mathcal{O}((\log \ell + \log M)\mathsf{poly}(\lambda))$ *where* $\ell = |R|$ *and* $M = |S|$*.*

Proof. The construction is the same as the one of Construction 2 where the standard WE scheme is replaced by our new compact WE scheme for $(\ell - 1, \ell)$ threshold conjunction languages of Construction 3.

The size of the signature in the Construction 2 is dominated by the size of the non-compact WE ciphertexts. By replacing it by the compact WE for threshold conjunction languages, it is easy to see that the resulting signature has size $\mathcal{O}((\log \ell + \log M)\mathsf{poly}(\lambda))$.

The properties of correctness, unforgeability and anonymity can be established using the same arguments as the ones of Thms. 4 to 6. □

Chapter 4

Practical Signature Witness Encryption and Time Release

In this chapter, we formally introduce the notion of Signature Witness Encryption (SWE) for multi-signatures, which allows users to encrypt a plaintext with respect to a tag and a set of signature verification keys. Once a multi-signature on the tag under a threshold number of the verification keys is released, this signature can be used to efficiently decrypt an SWE ciphertext for this tag.

We instantiate an SWE scheme for a modified BLS multi-signature scheme, focusing on concrete efficiency of our constructions. We demonstrate that our modified BLS scheme is in fact an *aggregate* signature, allowing to securely compress signatures of different signers on potentially different messages into an aggregated signature. While we will only define our SWE scheme with regard to multi-signatures, where all singers sign *the same* message, this is to show that our modified BLS can be used comparably to the original BLS signature scheme [BLS01], which is adapted to support aggregation of signatures on distinct messages [BGLS03] as well as multi-signatures [Bol03; RY07].

To showcase practicability of SWE, we further present McFly, which is a protocol that combines SWE with a suitable blockchain in order to achieve time-release encryption contingent on the blockchain committee signing blocks.

The chapter is taken almost verbatim from [DHMW22], which is the full version of [DHMW23]. These are joint work with Nico Döttling, Lucjan Hanzlik and Bernardo Magri published at Financial Cryptography and Data Security 2023.

4.1 Structure and Contributions

The structure of this chapter is as follows:

- In § 4.2 we will provide a technical outline detailing the high-level ideas of our construction as well as comparisons to related work and a discussion of our results.
- In § 4.3 we formally introduce the definition of signature-based witness encryption for multi-signatures and further state the definitions of aggregate and multi-signatures that we will work with in this chapter. Specifically, we assume a key registration model: Verification keys must be provided with a proof of knowledge of the corresponding secret key.
- Then we proceed to construct these primitives: We construct an aggregate/multi-signature scheme based on BLS in § 4.4, and a compatible SWE scheme in § 4.5.
- In § 4.6, we add a NIZK proof-system to show well-formedness and additional properties of encrypted messages and make our SWE scheme verifiable.
- In the end, we showcase practicability of our SWE scheme, by providing our McFly protocol, which integrates SWE with a blockchain and its properties

and possible applications in Section 4.7. This includes a formalization of our requirements on the underlying blockchain and the resulting security guarantees of our McFly protocol in § 4.7.1.

We now outline our contributions in a little more detail:

An Efficiency-Oriented Signature Witness Encryption Construction. Our construction of signature-based witness encryption (SWE) is made with efficient deployment in a blockchain context in mind. We focus here on the original definition of SWE, which originates from [DHMW22] and mandates compatibility of the scheme with multi-signature schemes as input and batch encryption for efficiency. We instantiate SWE with a multi-signature scheme $\mathsf{Sig}_{\mathsf{agg}}$, which is a BLS scheme with a modified aggregation mechanism. This choice is informed by BLS being a popular choice in blockchains due to their small signature size when aggregating. Our construction is based on the Boneh-Franklin IBE [BF01] and is proven secure in the random oracle model under the BDH assumption. As opposed to the Boneh-Franklin IBE, which is restricted to bit messages, we allow a message space of k-bit messages for a small fixed k.

Informal Theorem 5 (Constructions 4 and 5, Thms. 10 to 12 and 14)**.** *There exists a pairing-based signature witness encryption scheme* SWE *for a BLS-style multi-signature with modified aggregation, which is proven secure in the random oracle model under the BDH assumption and the co-CDH assumption. It supports batch encryption of multiple messages of k bits each. Encrypting ℓ messages with respect to n verification keys results in a ciphertext size of $\mathcal{O}(n + \ell)$.*

Concrete *efficiency* of encryption and decryption of SWE is supported by a reference implementation, targeting a closely related variant, which is presented in the full version of our paper [DHMW22]. A small discussion on efficiency can be found in § 4.5.2.

Verifiability. Our definition of SWE as described so far only provides the guarantee that *correctly* encrypted SWE ciphertexts allow decryption if and only if a corresponding multi-signature becomes available.

Towards use cases in decentralized settings with limited trust, we want to assure the receiver of an SWE ciphertext that decryption will work and yield a *useful* message. We add a NIZK proof layer, to make our SWE scheme verifiable in the sense that an encryptor can show that 1) decryption will consistently be possible, once a multi-signature under any qualified signer set becomes available and that 2) the encrypted message has additional properties captured by an NP language.

Informal Theorem 6 (Construction 7, Thm. 17)**.** *There exists a NIZK proof consisting of* (SWE.Prove, SWE.Vrfy) *that extends* SWE *to be a* verifiable *SWE scheme, which is proven secure in the random oracle model under the BDH assumption.*

McFly Protocol. As an application, we build an "encryption to the future" protocol, called McFly, by combining SWE with a Byzantine Fault Tolerance (BFT) blockchain or a blockchain finality layer.

In McFly, we will treat the blockchain as a time-keeper: We SWE-encrypt a message with respect to a specified *block height* (which represents how far into the future the message should remain hidden) and the set of verification keys of the committee members expected to sign that block. Once the committee produces and publishes the block at the specified height, the resulting block signature serves as the witness required to decrypt the SWE ciphertext.

We will provide some context on this protocol and the necessary properties of the underlying blockchain.

BFT Blockchains and Finality. In a BFT blockchain like Afgjort [DMMNT20], assuming an honest (super)majority of committee members, *finality* is guaranteed: once a block is agreed upon, it is irrevocably part of the blockchain, and no conflicting alternative branch (a *fork*) can later appear. In a finality layer, such as Ethereum's Casper [BG19], finality is applied only to some blocks at regular intervals, which serve as checkpoints. For simplicity, in this chapter, we will explicitly assume that all blocks are immediately finalized, but our results can be easily adapted to the more general setting where a subset of blocks at predictable heights are finalized.

This finality allows us to treat published blocks as reliable, immutable time markers for our time-release functionality McFly.

Blockchain Requirements. To instantiate McFly, the underlying BFT or finality layer blockchain must satisfy the following additional properties:

- **Known Committee.** Every (final) block in the chain must be signed by a committee of parties. We model these committees as static. Our protocol extends to the case of dynamic committees. In that case, the committee responsible for signing a block at a particular height must be known *some time* in advance. How much "time in advance" the committee is known is what we call the horizon (following the nomenclature of Gentry et al. [Gen+21]). This horizon limits how far into the future we can encrypt.
- **Multi-Signature Scheme.** Block signatures must be generated using an SWE-compatible (multi-)signature scheme.
- **Block Structure.** We assume that blocks have a predictable header, which we will model by a block counter, and some data content. When finalizing a block the committee signs the block as usual, but additionally, it also signs the block counter separately. They can use the same keys for this. This is safe whenever the underlying signature is a hash-and-sign scheme as is commonly the case.
- **Public Key Infrastructure.** The public keys of the committee members must have a proof of knowledge. This can be achieved, e.g., by registering the keys with a PKI.
- **Honest Majority Committee.**[1] The majority of the committee behaves honestly. That is, there will not be a majority of committee members colluding to prematurely sign blocks.
- **Constant Block Production Rate.** To have a meaningful notion of "wall-clock time", the blocks must be produced at a near constant rate.

To model the requirements above, we present a blockchain functionality in § 4.7.1 and later we show the security of the McFly protocol in this hybrid model.

Informal Theorem 7 (Construction 8, Thms. 18 to 20)**.** *We can integrate a blockchain with properties as listed above with* SWE *to achieve efficient time-release encryption, which is available for encryptors and decryptors with read-only access to the underlying blockchain. Data is encrypted to a future block height and decryption is possible only once the block at this height is published. The committee performs only blockchain maintenance and time-release encryption operates transparently on*

[1] The honest majority requirement must be strengthened to honest supermajority (i.e. at least 2/3 of members being honest) if the underlying blockchain or finality layer considers a partially synchronous network model. For simplicity, we choose to describe it in the synchronous network model where honest majority plus PKI is sufficient.

top of the blockchain. Security holds w.r.t. a static committee with honest majority and static corruption.

For concreteness, we informally discuss in § 4.7.4 how a modified version of the currently deployed Ethereum 2.0 running with Casper "The friendly finality gadget" [BG19] satisfies the requirements of our blockchain functionality. Intuitively, to make Ethereum 2.0 + Casper compatible with our blockchain model we only need to add the public-key-infrastructure and require the committee members to sign a block counter separately for each finalized block. This enables encryption up to the horizon where a future committee is already known. Unfortunately, in Ethereum 2.0 this leads to a maximum horizon of 12.8 minutes. If we use "sync committees" instead, that was only introduced in Ethereum Altair [But], we can have an horizon of up to 54 hours. However, it is not clear whether sync committees enjoy the same level of trust as the standard ones.

We discuss applications to frontrunning-prevention, decentralized auctions and randomness beacons in § 4.7.5.

Individual Contribution. The beginnings of this project stem from two different directions: The ideas culminating in the McFly protocol and blockchain integration came out of research meetings that Bernardo Magri and Nico Döttling had. In the mean time, the concept of Signature Witness Encryption (SWE) emerged from a series of discussions between Nico Döttling and myself concerning blockchain-based protocol design. Nico developed the first construction during one of our whiteboard sessions. He then brought these two lines of discussion together and initiated a collaboration between him, Bernardo Magri, Lucjan Hanzlik and me. Bernardo and Lucjan contributed further perspectives on open problems and practical needs in the blockchain space, informing the design direction for both SWE and the McFly protocol. One contribution by Lucjan, specifically, was the observation that towards a practical solution it would be necessary to construct SWE in a way that mainly requires computation in one of the groups while minimizing the computations in the other two groups.

The Introduction of the paper was written collaboratively. The Technical Overview was primarily drafted by Nico, with some input from me. I contributed the formal definitions and wrote most of the section describing the modified BLS construction (Construction 4). Following Nico's outline, I produced an initial write-up of the SWE construction, which was later optimized following Lucjans observations and joint brainstorming. The current version was further improved and written up by Nico shortly before submission to improve efficiency by pushing operations into the source group—resulting in Construction 6 (SWE'). The proof of IND-CPA security in its current form for SWE' was developed largely by Nico.

After I identified a critical issue in the construction, namely security breaking down when reference messages are not distinct, I raised this with Nico and we worked together to develop a minimal fix addressing the problem. This led to the conception of the improved scheme SWE given in Construction 5, which I then integrated into the paper, adding the construction, developing security proofs, and updating dependencies accordingly.

Regarding the NIZK proofs, the protocols and initial proof sketches were written by Nico. I later extended them with detailed arguments, adding structure and rigor in response to reviewer feedback. The current write-up in this thesis further improves on the original version by clarifying details and making the arguments more formally precise.

The McFly section was mostly written by Bernardo and me, with some input from Lucjan and Nico. Specifically, the blockchain model, protocol guarantees, description,

and proof of our McFly Construction 8 were mostly written by me, while details about blockchain integration and applications were mostly written by Bernardo.

Lucjan developed the reference implementation of the SWE' scheme, which is presented in the full version of the paper.

4.2 Technical Overview

The key ingredient and main technical challenge of this chapter is *Signature Witness Encryption* (SWE). In the following, we will provide an outline of our construction of practically efficient SWE.

4.2.1 Developing Efficient SWE for BLS Multi-Signatures

SWE Based on BLS. Our construction of Signature-based Witness Encryption is based on the BLS signature scheme [BLS01] and its relation to identity-based encryption [BF01]. Recall that BLS signatures are defined over a bilinear group, i.e. we have three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ (with generators g_1, g_2, g_T) of prime-order p and an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. A verification key vk is of the form $\mathsf{vk} = g_2^x$, where $x \in \mathbb{Z}_p$ is the corresponding signing key. To sign a message $T \in \{0,1\}^*$, we compute $\sigma = H(T)^x$, where $H : \{0,1\}^* \to \mathbb{G}_1$ is a hash function (which is modeled as a random oracle in the security proofs). To verify a signature σ for a message T, all we need to do is check whether $e(\sigma, g_2) = e(H(T), \mathsf{vk})$. The BLS signature scheme is closely related to the identity-based encryption scheme of Boneh and Franklin [BF01].[2] Specifically, in the IBE scheme of [BF01] BLS verification keys take the role of the master public key, the signing key takes the role of the master secret key and signatures take the role of identity secret keys, where the signed messages correspond to the identities, respectively. In this sense, the BF scheme can be seen as a witness encryption scheme that allows to encrypt plaintexts m with respect to a verification key vk and a message T, such that anyone in possession of a valid signature of T under vk will be able to decrypt the plaintext m. Specifically, we can encrypt a message $m \in \{0,1\}$ by computing $\mathsf{ct} = (g_2^r, e(H(T), \mathsf{vk})^r \cdot g_T^m)$. Given a signature $\sigma = H(T)^x$, we can decrypt a ciphertext $\mathsf{ct} = (c_1, c_2)$ by computing $d = c_2/e(\sigma, c_1)$ and taking the discrete logarithm of d with respect to g_T (which can be done efficiently as $m \in \{0,1\}$).

SWE for BLS Multi-Signatures. The BLS scheme can be instantiated as a multi-signature scheme [BGLS03; Bol03; RY07]. Specifically, assume that for $i = 1, \ldots, n$ we have signatures σ_i with respect to a verification key vk_i, which all verify for a message T. Then we can combine the signatures $\sigma_1, \ldots, \sigma_n$ into a *single* compact signature $\sigma = \prod_{i=1}^n \sigma_i$. Verifying such a signature can be done by checking whether $e(\sigma, g_2) = \prod_{i=1}^n e(H(T), \mathsf{vk}_i)$, where correctness follows routinely. We can adapt the BF IBE scheme to multi-signatures in a natural way: To encrypt a plaintext $m \in \{0,1\}$ to a reference message T and verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$, we compute a ciphertext ct via $\mathsf{ct} = (c_1 = g_2^r, c_2 = (\prod_{i=1}^n e(H(T), \mathsf{vk}_i))^r \cdot g_T^m)$. We note, that BLS is also extended to be an aggregate signature [BGLS03] and that it is also possible to modify the Boneh-Franklin scheme to encrypt to different messages $T_1, \ldots, T_n$ for each individual signer, but this is not necessary for our envisioned use case and will

[2] In fact, in [BF01] Boneh and Franklin already mention that there is a general transformation from an IBE scheme to a signature scheme, which has been informally observed by Moni Naor. The relationship we describe between the BF-IBE and BLS signatures generally holds for signatures that have been received from an IBE scheme by this Naor-transform and thus, in principle, a basic SWE scheme follows directly from any IBE.

not be possible in our final construction due to technical reasons, so we focus on the multi-signature case here.

Such a ciphertext $\mathsf{ct} = (c_1, c_2)$ can be decrypted analogously to the above by computing $d = c_2/e(\sigma, c_1)$ and taking the discrete logarithm with respect to g_T. To decrypt ct we need a multi-signature σ of T under *all* the verification keys vk_i. For our envisioned applications this requirement is too strong, instead, we need a *threshold* scheme where a t-out-of-n multi-signature suffices as a witness to decrypt a ciphertext. Thus, we will rely on Shamir's secret sharing scheme [Sha79] to implement a t-out-of-n access structure. This, however, leads to additional challenges. Recall that Shamir's secret sharing scheme allows us to share a message $r_0 \in \mathbb{Z}_p$ into shares $s_1, \ldots, s_n \in \mathbb{Z}_p$, such that r_0 can be reconstructed via a (public) linear combination of any t of the s_i, while on the other hand, any set of less than t shares s_i reveals no information about r_0. The coefficients L_{i_j} of the linear combination required to reconstruct r_0 from a set of shares $s_{i_1}, \ldots, s_{i_t}$ (for indices $i_1, \ldots, i_t$) can be obtained from a corresponding set of Lagrange polynomials. Given such L_{i_j}, we can express r_0 as $r_0 = \sum_{j=1}^{t} L_{i_j} s_{i_j}$. We can now modify the above SWE scheme for multi-signatures as follows. To encrypt a plaintext $m \in \{0,1\}$, we first compute a t-out-of-n secret sharing $s_1, \ldots, s_n$ of the plaintext m. The ciphertext ct is then computed by $\mathsf{ct} = (g_2^r, (e(H(T), \mathsf{vk}_i)^r \cdot g_T^{s_i})_{i\in[n]})$. Security of this scheme can be established from the same assumption as the BF IBE scheme, namely from the bilinear Diffie-Hellman (BDH) assumption, as introduced in the context of tripartite Diffie Hellman by Joux [Jou00] and formalized in [BF01]. We would now like to be able to decrypt such a ciphertext using a multi-signature. For this purpose, however, we will have to modify the aggregation procedure of the multi-signature scheme. Say we obtain t-out-of-n signatures σ_{i_j}, where σ_{i_j} is a signature of T under vk_{i_j}. Let L_{i_j} be the corresponding Lagrange coefficients. Our new aggregation procedure computes $\sigma = \prod_{j=1}^{t} \sigma_{i_j}^{L_{i_j}}$. That is, instead of merely taking the product of the σ_{i_j} we need to raise each σ_{i_j} to the power of its corresponding Lagrange coefficient L_{i_j}. We can show that this modification does not hurt the security of the underlying BLS multi-signature scheme. To decrypt a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_n)$ using such a multi-signature σ, we compute $d = \prod_{j=1}^{t} c_{i_j}^{L_{i_j}}/e(\sigma, c_0)$ and take the discrete logarithm of d with respect to g_T. Correctness follows via a routine calculation.

Efficiency-Boosting Design Choices. While the above scheme provides our desired functionality, implementing this scheme leads to a very poor performance profile. There are two main reasons: (1) Each ciphertext encrypts just a single bit. Thus, to encrypt any meaningful number of bits we need to provide a large number of ciphertexts. Observe that each ciphertext contains more than n group elements. Thus, encrypting k bits would require a ciphertext comprising kn group elements, which would be prohibitively large even for moderate values of k and n. (2) Both encryption and decryption rely heavily on pairing operations and operations in the target group. From an implementation perspective, pairing operations and operations in the target group are typically several times slower than operations in one of the source groups.

To address these issues, we will design a scheme that allows for *ciphertext packing* as well as *batch encryption*, allowing to encrypt multiple k-bit messages m_i with respect to reference messages T_i in one go. The scheme further shifts almost all group operations into one of the two source groups (in our case this will be $\mathbb{G}_2$). In § 4.5.1 we provide two variations of this construction, SWE' and SWE, where SWE' provides slightly smaller ciphertexts, but is only secure when the reference messages T_i are distinct. We will highlight here only a few core design aspects central to both of those constructions:

- Instead of just encrypting single bits $m_i \in \{0,1\}$, we allow each message m_i to come from $\{0,\ldots,2^k-1\}$. This will allow us to pack k bits into each ciphertext component. Recall that decryption requires the computation of a discrete logarithm with respect to a generator g_T. We can speed up this computation by relying on the Baby-Step-Giant-Step (BSGS) algorithm [Sha71] to $O(2^{k/2})$ group operations. An appropriate choice of k leads to a very efficient implementation as the required discrete logarithm table for the fixed generator g_T can be precomputed.
- Instead of computing a secret sharing of each plaintext m_i, we compute a secret sharing of a random value $r_0 \in \mathbb{Z}_p$. The value r_0 can be used to randomize many batch-ciphertext components, leading to ciphertexts consisting of only $O(\ell+n)$ group elements for encrypting ℓ k-bit messages m_i with respect to n verification keys.
- We encrypt each share s_i in the source group $\mathbb{G}_2$ instead of $\mathbb{G}_T$. That is, we compute the ciphertext-component c_i via $c_i = \mathsf{vk}_i^r \cdot g_2^{s_i}$. This necessitates a corresponding modification of the decryption algorithm and will be the reason that our scheme only works with multi-signatures of a single reference message T_i per encrypted m_i, as opposed to one potentially different reference message for each signer. Somewhat surprisingly, this modification does not necessitate making a stronger hardness assumption, but only requires a rather intricate random-self-reduction procedure in the security proof. That is, even with this modification we can still rely on the hardness of the standard BDH assumption.

4.2.2 A Compatibility-Layer for Efficient Proof Systems

Our scheme so far assumes that encryptors behave honestly, i.e., the ciphertext ct is well-formed. A malicious encryptor, however, may provide ciphertexts that do not decrypt consistently, i.e. the decrypted plaintext m may depend on the signature σ used for decryption. Furthermore, for several of the use cases, we envision it is crucial to ensure that the encrypted message m satisfies additional properties. To facilitate this, we provide the following augmentations.

- We provide an efficient NIZK proof in the ROM which ensures that ciphertexts decrypt consistently, i.e. the result of decryption does not depend on the signature which is used for decryption. This is provided in § 4.6.1.
- We augment ciphertexts with efficient *proof-system enabled commitments* and provide very efficient plaintext equality proofs in the ROM. In essence, we provide an efficient NIZK proof system that allows to prove that a ciphertext ct and a Pedersen commitment C commit to the same value. This is discussed in § 4.6.2.
- We can now rely on Bulletproofs [Bün+18], an efficient and succinct proof system, in order to establish additional guarantees about the encrypted plaintext. For instance, we can rely on the range-proofs of [Bün+18] to ensure that the encrypted messages are within a certain range to ensure that our BSGS decryption procedure will recover the correct plaintext efficiently. This is discussed in § 4.6.3.

To make this construction efficient, we include redundancy into our SWE ciphertexts, effectively making ciphertexts homomorphic commitments.

4.2.3 Related Work

There is a rich body of works touching on similar topics. A comprehensive summary and comparison are provided below.

BLS Signatures. The BLS signature scheme, introduced in [BLS01] by Boneh, Lynn, and Shacham, is a pairing-based signature scheme with signatures of one group element in size. Due to their small size and very space-efficient aggregation, BLS signatures are used in widely deployed systems such as Ethereum 2.0 [eth22]. At first, BLS was extended to an aggregate signature in [BGLS03], where signatures of different users on different messages can be compressed into a single signature, thus saving space. The need for distinct messages stemmed from a security concern dubbed *rogue-key attacks*, where an adversary can maliciously choose verification keys adaptively based on honest parties' keys such that the honest parties' contribution(s) cancel out in aggregate signature verification, enabling forgeries of multi-signatures. For example for key vk, an adversary might choose $\mathsf{vk}^* = g_2^r/\mathsf{vk}$ and create a multi-signature for any message T under $\mathsf{vk}, \mathsf{vk}^*$ as $H(T)^r$ without requiring knowledge of the secret key for vk.

Multi-signature support for BLS was nevertheless achieved in [Bol03; RY07] under the assumption that signers must register their verification keys with a key registry, which requires them to provide a proof of knowledge or proof of possession of the underlying secret key. Another alternative, which was already pointed out in [BGLS03] is to *enforce* distinctness of messages by signing not the hash of just the message, but the hash of the signer verification key *and* the message. The analysis of this approach was refined in [BNN07], where Belare, Namprempre and Neven show that this method enables aggregation for arbitrary (different or duplicate) messages and signers.

Another approach to combine aggregate and multi-signing features are *aggregate multi-signatures* as introduced in [BDN18]. There, multiple multi-signatures on different messages can be aggregated into one aggregate multi-signature without requiring a proof of knowledge or possession. They require however, that partial signatures and aggregation for the underlying multi-signature scheme depend on the concrete multi-signer, so all signatures have to be combined in one step.

Our signature $\mathsf{Sig}_{\mathsf{agg}}$ suffers from the same drawback, as it can only aggregate once, as our aggregation similarly depends on the signer set. As outlined before, we need to raise individual signatures to the power of corresponding Lagrange coefficients, to be compatible with our SWE scheme. These Lagrange coefficients depend on which parties participate by submitting a signature. Further, $\mathsf{Sig}_{\mathsf{agg}}$ relies on users providing an online-extractable proof of knowledge. While we will show that our construction $\mathsf{Sig}_{\mathsf{agg}}$ achieves unforgeability in the sense of [BNN07], allowing arbitrary signer-message combinations, we note that it suffers from both of the major restrictions in the literature (one-time aggregation and key registration).

Signature Witness Encryption and Comparable Notions. At the time of writing our paper [DHMW22], the notion of SWE was novel, however we identify three types of pre-existing notions that can be seen as related: identity-based encryption (IBE), threshold encryption and witness encryption (WE).

Identity-based encryption was first introduced by Shamir [Sha84]. The initial idea was to use the identity – e.g. a mailing address – as a public key that messages can be encrypted to. *Threshold encryption schemes* [DF90; Fra90] are encryption schemes where the decryption key is distributed among a number of decryption servers, such that a threshold number of those servers needs to participate to facilitate decryption. In the classic setting, this set of servers is fixed at setup, and decryption is performed by a central entity, the combiner, that interacts with the underlying servers to collect and combine partial decryptions.

In a sense, our scheme can be seen as a threshold IBE, as we encrypt with respect to a committee and can only decrypt if a threshold of the committee members collaborate. In the light of this, we note again, that our construction is to some degree based on combining the Boneh-Franklin IBE [BF01] with the secret sharing scheme due to Shamir [Sha79], and a basic threshold SWE scheme without additional performance optimizations or verifiability may be received from any IBE via the Naor-transform plus secret sharing.

However, our notion aims at a less interactive goal than threshold decryption. We do not wish for a joint setup being necessary for the parties involved in decryption, neither do we wish for active participation beyond providing signatures. In a setting where we expect signatures on predictable outcomes to appear naturally, SWE can be run *transparently*, without the signing parties being aware that they are aiding in decryption.

Another way to see SWE conceptually is as a witness encryption scheme with a signature as a witness. We recall again, that *Witness Encryption* [GGSW13] is defined for an NP language $\mathcal{L}$ with poly-time relation $\mathcal{R}$. Encryption of a message m is w.r.t. a statement x and decryption is possible with a witness w s.t. $(x, w) \in \mathcal{R}$. Security intuitively says that a ciphertext does not reveal any information about m if $x \notin \mathcal{L}$. Security is extended in [GKPVZ13] in the form of *extractable* witness encryption, which also offers security guarantees even when $x \in \mathcal{L}$. Unlike standard witness encryption, which is only known from strong assumptions and computationally expensive, our SWE scheme is concretely efficient, but targets a less expressive release condition.

Concurrent Work. We would like to highlight two papers, that share similar constructions/techniques to our main SWE construction and can be considered concurrent. Both of them also rely heavily on BLS signatures and the Boneh-Franklin IBE scheme conceptually.

In [GMR23], Gailly, Melissaris, and Romailler achieve better scalability into the future by running a very similar system as our SWE scheme, but not with respect to a generic blockchain. Instead, it is run with the drand randomness beacon in mind, which publishes BLS signatures on predictable messages as random values. Its advantage for deployment of an SWE-like scheme is, that the public key of the committee remains constant. Encryption takes place with this public key as a single verification key. The drawback of this solution is, that drand requires a distributed key generation protocol between beacon maintainers and the committee is relatively static. This means that this protocol achieves better scalability of possible encryption times in practice, but also relies more concretly on trust in the institutions running the drand network.

In [CCNPS23], Cerulli et al. use similar techniques to ours, but the focus is more on key derivation in the future with the help of third parties, whereas our paper focuses on sending messages into the future without third parties.

There has been additional follow-up work, including our work on compact SWE in the standard model [ADMSW24], which Chapter 5 of this thesis is based on. To keep duplicate discussions minimal, we refer the reader to the related work section § 5.2.2 contained in that chapter for a discussion of further recent advancements in this area.

Classical Time-Release Cryptography Compared to McFly. As one of the main motivations for our initial introduction of SWE was time-release, let us compare where our resulting McFly protocol fits into classical notions of time-release.

The notion of timed-release encryption was proposed in the seminal paper by Rivest, Shamir and Wagner [RSW96]. The goal is to encrypt a message so that it cannot be decrypted, not even by the sender, until a pre-determined amount of time has passed. This allows to "encrypt messages to the future". In [RSW96] the authors propose two orthogonal directions for realizing such a primitive. Using trusted third-parties to hold the secrets and only reveal them once the pre-determined amount of time has passed, or by using so-called time-lock puzzles, that are computational problems that can not be solved without running a computer continuously for at least a certain amount of time.

An interesting example of the latter are timed commitments [BN00], which are commitments with an additional forced opening phase that requires a specified (big) amount of computation time. This is useful in an optimistic setting, where cooperation is usually the case, as an honest party can convince the receiver of the committed value without needing to do the timely decryption step. This is indeed also possible for our SWE scheme, as we discuss in § 4.6, that ciphertexts constitute a statistically binding commitment, but that is not our focus, as our decryption is efficient enough to be run. In case of one party aborting, timed commitments share all drawbacks of time-lock puzzles, whereas our protocol works efficiently, even if the encryptor only submits their value and then goes offline.

Our approach in McFly using SWE is more closely related to the paradigm of using a trusted party as in [CHKO08; CLQ05]. Simply put, these approaches set up a dedicated server that outputs tokens for decryption at specified times. We could deploy SWE in such a scenario as well, with the tokens being multi-signatures on predictable messages. Specifically, both Cathalo, Libert, and Quisquater [CLQ05] and our scheme achieve that no communication needs to take place between the trusted server and other entities. However, complete trust in a single (or multiple) server is a strong assumption, thus we re-use the decentralized architecture, computation and trust structure already present in blockchains.

Time-Release Powered by Blockchains. With the advent of blockchains, multiple proposals to realize timed-release encryption using the blockchain as a time-keeping tool emerged already. The previous results, presented next, are primarily of theoretical interest, whereas the construction developed in this thesis has been shown to be practically efficient through an implementation reported in our paper [DHMW23].

In [LJKW18], Liu, Jager, Kakvi, and Warinschi propose a scheme based on extractable witness encryption using the blockchain as a reference clock; messages are encrypted to future blocks of the chain that, once created, can be used as a witness for decryption. However, as we already noted, extractable witness encryption is a very expensive primitive.

An area of study surrounding *committee-based witness encryption* emerged, where witness encryption is achieved with the help of a (blockchain) committee, under threshold trust assumptions.

In [GKMPS22] Goyal et al. introduce the notion of extractable witness encryption on the blockchain (eWEB). In this scheme, a message encrypted with respect to a statement x is secret-shared among several committee members and labeled with x. The decryptor must interact with a threshold number of committee members, proving they have a valid witness to collect the shares and decrypt the message. However, the storage complexity for each committee member increases with the number of shares they hold. This limitation is addressed by Erwig, Faust, and Riahi [EFR21], which proposes creating a joint public key for encryption while giving each committee member a share of a correlated secret key. Another work by Faust, Hazay, Kretzler,

and Schlosser [FHKS23] strengthens this approach by also hiding the underlying statement of the encryption. Their ciphertext size is independent of the number of keys. However, they require a correlated key setup where the committee has a single public key and the secret key is shared among the members.

Concurrently to our paper [DHMW23], Campanelli et al. [CDKKN22] also proposed an "encryption to the future" scheme based on proof-of-stake blockchains. Their version of committee-based witness encryption relies on deploying a smart contract on the blockchain for each encryption. Their encryption to the future approach is geared toward transmitting messages from past committee members to future slot winners of the proof-of-stake lottery and requires active participation in the protocol by the committee members.

The critical conceptual difference between SWE and previous works on committee-based witness encryption is that in SWE committee keys are independently sampled, and decryption relies solely on the availability of signatures from the committee. This means that our McFly scheme offers encryption to the future even for encryptors and decryptors that only read the state of the blockchain and we require no active participation of the committee beyond their regular duties, assuming, that predictable messages like a block header are already signed in each (finalized) block. Otherwise, all committees need to only include this one additional signature, irrespective of pending timed encryptions, so there is no direct involvement between users of McFly and committees.

Another related line of work is presented by Benhamouda et al. [Ben+20], where a message is kept secret and "alive" on the chain by re-sharing a secret sharing of the message from committee to committee. This allows to keep the message secret until an arbitrary condition is met and the committee can reveal the message. A more general approach is the recent YOSO protocol [Gen+21] that allows to perform secure computation in that same setting, by using an additive homomorphic encryption scheme, to which committees hold shares of a secret key and continuously re-share it. While these approaches realize some form of encryption to the future, they require massive communication from parties and are still far from practical.

Further, the idea of using threshold encryption directly [DFL24; MGZ23] as a tool to "encrypt transactions to the next slot" was introduced concurrently to our work in the context of frontrunning protection. These works employ threshold encryption using a dedicated decryption committee (with distributed key generation): Transactions are encrypted by users to the next block and once that block is finalized, the decryption committee releases information that allows to decrypt all transactions in the block.

A spin on timed commitments is also available using blockchains; in [ADMM14], Andrychowicz, Dziembowski, Malinowski, and Mazurek introduce a blockchain contract that locks assets of the commitment sender for a set time based on a commitment. If the sender fails to open the commitment within that time, their assets are made available to the receiver as a penalty; however, the commitment is not opened in that case.

4.2.4 Discussion and Interpretation

As demonstrated by our own results and related works, it is practically feasible to implement signature witness encryption, which enables conditional release based on the availability of (a threshold number of) signatures on predictable messages. This type of construction can be used to implement efficient timed release on the blockchain, especially when paired with a committee using distributed key generation as suggested in [GMR23].

There are however, some limitations in current proposals, that we would like to overcome in the future: Firstly, current constructions target signature schemes that are retrieved from IBE with the Naor-transform with concretely efficient optimizations developed specifically for BLS signatures. It would be interesting to provide practical alternatives for other practically used signature schemes. From a theoretical point of view this study could be worthwhile, as it essentially relates to the study of the relationship of signatures and IBE schemes itself beyond the Naor-transform.

Secondly, we noticed that *known* messages must be somewhat artificially enforced in our use case. In fact, for the implementation of [GMR23], a new unchained mode was added to drand to provide signatures of known messages as random beacon outputs. We can envision that there are blockchain-based applications where a more *flexible* approach could be beneficial, where signature(s) can be used to decrypt if they are on some message m', which fulfills a *predicate* which is determined at decryption time. For example, in fair exchange, we might want to encrypt to a committee's signature on a transaction paying us a fixed amount. Such a transaction itself consists of a signature under the sender's key and may not be known at encryption time. Allowing arbitrary predicates seems likely to give rise to general witness encryption, as NP witnesses could be required as part of the predicate, but the study of less complex predicates could still expand use cases.

Thirdly, from a theoretical point of view, it would be satisfying to investigate whether modifications can be made to McFly and the underlying blockchain to enable transparent encryption further into the future, even in a setting with permissionless committees. Quintessentially asking whether we can *re-encrypt* SWE ciphertexts to future committees with no or very limited additional work done by the committee (e.g. publishing a re-encryption key once when handing off to a new committee).

4.3 Definitions

We will introduce the definition of signature witness encryption in this section. Its underlying multi-signature scheme will be based on an *aggregate signature* scheme in a key registration model, where verification keys come with a proof of knowledge. We will first state the relevant signature definitions before we move on to defining SWE.

4.3.1 Aggregate Signatures and Multi-Signatures

Aggregate signatures and multi-signatures are digital signatures that allow to compress multiple signatures into one. Multi-signatures allow this for multiple users signing the same message, while aggregate signatures allow to combine signatures from multiple users on multiple messages that may contain duplicates into one aggregate signature.[3] We will define aggregate signatures first and then position multi-signatures as a special case.

For technical reasons in the remainder of the chapter, we need aggregatable signature schemes for which we can extract the secret keys in our reductions. Thus, we require all published public keys to come with an online-extractable proof of knowledge that shows, that the issuer knows a corresponding secret key. This is similar to the multi-signature based on BLS constructed in [RY07] requiring all public keys to be registered with a key registry via a proof of possession. Such a proof can either

[3] While aggregate signatures were introduced for *distinct* messages and signers in [BGLS03], we use a notion similar to *unrestricted aggreggate signatures* as introduced in [BNN07], which work for arbitrary signer-message pairs.

be appended to one's public verification key or registered with a certifying authority/key registry and allows us to extract secret keys in our proofs. All algorithms below implicitly have access to a random oracle H.

Definition 63 (Aggregate Signatures)**.** *An aggregate signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Agg}, \mathsf{AggVrfy}, \mathsf{Prove}, \mathsf{Valid})$ *is a tuple of seven algorithms where:*

- $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*: The key generation algorithm takes a security parameter and outputs a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sk})$*.*
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, T)$*: The signing algorithm takes as input a signing key* sk *and a message* T*. It outputs a signature* σ*.*
- $b \leftarrow \mathsf{Vrfy}(\mathsf{vk}, T, \sigma)$*: The verification algorithm takes as input a verification key* vk*, a message* T *and a signature* σ*. It outputs a bit* b*.*
- $\sigma \leftarrow \mathsf{Agg}((\sigma_1, \ldots, \sigma_k), (\mathsf{vk}_1, \ldots, \mathsf{vk}_k))$*: The aggregation algorithm takes a list of signatures* $(\sigma_1, \ldots, \sigma_k)$ *and verification keys* $(\mathsf{vk}_1, \ldots, \mathsf{vk}_k)$*. It outputs one aggregate signature* σ*.*
- $b \leftarrow \mathsf{AggVrfy}(\sigma, (\mathsf{vk}_1, \ldots, \mathsf{vk}_k), (T_1, \ldots, T_k))$*: The verification algorithm for aggregate signatures takes an aggregate signature* σ *as well as two lists of public verification keys* vk_i *and messages* T_i*, which may include duplicates. It outputs a bit* b*. For convenience, we consider this identical to* Vrfy*, if only* σ *and one key* vk_1 *and message* T_1 *are input.*
- $\pi \leftarrow \mathsf{Prove}(\mathsf{vk}, \mathsf{sk})$*: The proving algorithm has access to an additional hash function oracle* H_{pr}*, takes a verification key* vk *and a signing key* sk*. It outputs a proof* π*.*
- $b \leftarrow \mathsf{Valid}(\mathsf{vk}, \pi)$*: The validity algorithm has access to to an additional hash function oracle* H_{pr}*, takes a verification key* vk *and a proof* π*. It outputs a bit* b*.*

We require such a signature scheme to be correct and unforgeable and that $(\mathsf{Prove}, \mathsf{Valid})$ *is an online-extractable zero-knowledge proof of knowledge for the relation*

$$\mathcal{K} = \{(\mathsf{vk}, \mathsf{sk}) | \exists r \text{ s.t. } (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda; r).\}$$

Note that this implicitly requires that it is efficiently checkable, whether a secret key belongs to a given public key. This will be true for our use case.

Definition 64 (Correctness)**.** *An aggregate scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Agg}, \mathsf{AggVrfy}, \mathsf{Prove}, \mathsf{Valid})$ *is* correct *if for all* $\lambda \in \mathbb{N}$*,* $k = \mathsf{poly}(\lambda)$*, messages* $T, T_1, \ldots, T_k$*, sets of public keys* $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_k)$ *and verifying signatures* $(\sigma_1, \ldots, \sigma_k)$ *such that* $\mathsf{Vrfy}(\mathsf{vk}_i, T_i, \sigma_i) = 1$ *for* $i \in [k]$ *it holds:*

$$\Pr\left[\begin{array}{l} \mathsf{Vrfy}(\mathsf{vk}, T, \mathsf{Sign}(\mathsf{sk}, T)) = 1 : \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \end{array}\right] = 1$$

and

$$\Pr\left[\begin{array}{l} \mathsf{AggVrfy}(\sigma, V, (T_1, \ldots, T_k)) = 1 : \\ \sigma \leftarrow \mathsf{Agg}((\sigma_1, \ldots, \sigma_k), V) \end{array}\right] = 1.$$

Definition 65 (Unforgeability)**.** *An aggregate scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Agg}, \mathsf{AggVrfy}, \mathsf{Prove}, \mathsf{Valid})$ *is* unforgeable *if for all* $\lambda \in \mathbb{N}$*,* $N = poly(\lambda)$*, there is no PPT adversary* $\mathcal{A}$ *with more than negligible advantage in the experiment* $\mathsf{Exp}_{\mathsf{Unf\text{-}Agg}}(\mathcal{A}, N)$ *defined as follows:*

1. *The experiment randomly chooses* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and sends* vk *to* $\mathcal{A}$*.*

2. *$\mathcal{A}$ may request signatures from a signing oracle* $\mathsf{Sign}(\mathsf{sk}, \cdot)$.
3. *$\mathcal{A}$ may also request to see a proof for* vk*, which the experiment answers with* $\pi \leftarrow \mathsf{Prove}(\mathsf{vk}, \mathsf{sk})$*. If it does, $\mathcal{A}$ may not use* vk *as one of the keys in its forgery.*
4. *Eventually, $\mathcal{A}$ outputs* $((T_1, \ldots, T_k), (\mathsf{vk}_2, \ldots, \mathsf{vk}_k), (\pi_2, \ldots, \pi_k), \sigma^*)$ *as a forgery. If $\mathcal{A}$ requested a proof for* vk *before, but one of the* vk_i *for* $i \in \{2, \ldots, k\}$ *is* vk*, the experiment outputs* 0*. The case* $k = 1$*, where $\mathcal{A}$ outputs no additional keys is explicitly allowed.*
5. *The experiment outputs* 1*, if* $\mathsf{AggVrfy}(\sigma^*, (\mathsf{vk}, \mathsf{vk}_2, \ldots, \mathsf{vk}_k), (T_1, \ldots, T_k)) = 1$*,* T_1 *was never queried to the oracle* $\mathsf{Sign}(\mathsf{sk}, \cdot)$*,* $\mathsf{Valid}(\mathsf{vk}_i, \pi_i) = 1$ *for all* $i \in \{2, \ldots, k\}$ *and the number of keys* k *is upper bounded by* N*. Otherwise, the output is 0.*

We define the advantage in this experiment as $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Unf\text{-}Agg}} = \Pr\left[\mathsf{Exp}_{\mathsf{Unf\text{-}Agg}}(\mathcal{A}, N) = 1\right]$.

We define multi-signatures as a special case of these aggregate signatures.

Definition 66 (Multi-Signatures)**.** *A multi-signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Agg}, \mathsf{AggVrfy}, \mathsf{Prove}, \mathsf{Valid})$ *is a tuple of seven algorithms, analogously defined as in Def. 63 for aggregate signatures, with one difference: Signature aggregation* $\mathsf{AggVrfy}(\sigma, (\mathsf{vk}_1, \ldots, \mathsf{vk}_k), (T_1, \ldots, T_k))$ *is only defined for* $T_1 = T_2 = \cdots = T_k$*. Correct aggregation is only necessary to hold for aggregation across signatures of the same message. Similarly, the unforgeability experiment requires a valid forgery* $((T_1, \ldots, T_k), (\mathsf{vk}_2, \ldots, \mathsf{vk}_k), (\pi_2, \ldots, \pi_k), \sigma^*)$ *to additionally fulfill* $T_1 = \cdots = T_k$*.*

To ease readability, we will sometimes write $\mathsf{AggVrfy}(\sigma, (\mathsf{vk}_1, \ldots, \mathsf{vk}_k), T)$ instead of $\mathsf{AggVrfy}(\sigma, (\mathsf{vk}_1, \ldots, \mathsf{vk}_k), (T, \ldots, T))$ when working with multi-signatures.

4.3.2 Signature Witness Encryption for Multi-Signatures

In this section, we introduce the new cryptographic primitive SWE that is the core technical component of the McFly protocol. We formally define it next. All algorithms $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Vrfy})$ implicitly have access to a random oracle H.

Definition 67 (Signature-Based Witness Encryption)**.** *A t-out-of-n SWE for a multi-signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Agg}, \mathsf{AggVrfy}, \mathsf{Prove}, \mathsf{Valid})$ *is a tuple of two algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *where:*

- $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n), (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]})$*: Encryption takes as input a set* V *of* n *verification keys of the underlying scheme* Sig*, a list of reference signing messages* T_i *and a list of messages* m_i *of arbitrary length* $\ell \in \mathsf{poly}(\lambda)$*. It outputs a ciphertext* ct*.*
- $m \leftarrow \mathsf{Dec}(\mathsf{ct}, (\sigma_i)_{i\in[\ell]}, U, V)$*: Decryption takes as input a ciphertext* ct*, a list of multi-signatures* $(\sigma_i)_{i\in[\ell]}$ *and two sets* U, V *of verification keys of the underlying scheme* Sig*. It outputs a message composed of multiple parts* $m = (m_i)_{i\in[\ell]}$*.*

We require such an SWE scheme to fulfill robust correctness and IND-CPA security.

The idea of our security definitions is to model fine-grained access and batching; We consider ℓ messages $(m_i)_{i\in[\ell]}$ to be encrypted at once under respective reference messages $(T_i)_{i\in[\ell]}$. We want decryption of a specific m_{ind} to be possible if and only if we get a multi-signature of the corresponding T_{ind} under at least t keys. To model this, in our security definition, we will introduce a set $SC \subseteq [\ell]$ which marks the *challenge slots*, where the adversary will embed IND-CPA challenges. The adversary receives $n - t + 1$ honest verification keys (including a proof of knowledge) and may then adaptively select and register $t - 1$ corrupted keys. The adversary can ask for signatures under honest keys, and make an IND-CPA style query, where it can specify

the reference messages T_i for all slots, fixed messages m_i for the non-challenge slots $i \in [\ell] \setminus SC$ and challenge messages m_i^0, m_i^1 for the challenge slots $i \in SC$. The adversary may never query for a signature of a reference message that it uses for a challenge slot.

Remark on Message Space. In our construction, we will explicitly restrict the message-space: for all $i \in [\ell]$, $m_i \in \{0, \ldots, 2^k - 1\}$ must hold for a small integer k, so that discrete logarithms can be computed efficiently. For generality of our definition, we do not enforce this restriction in the syntax of Def. 67, but it is necessary for practical use of our scheme and informs our definition of verifiability later.

Definition 68 (Robust Correctness)**.** *A t-out-of-n SWE scheme* $\mathsf{SWE}' = (\mathsf{Enc}, \mathsf{Dec})$ *for a multi-signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Agg}, \mathsf{AggVrfy}, \mathsf{Prove}, \mathsf{Valid})$ *is* correct *if for all* $\lambda \in \mathbb{N}$ *and* $\ell = \mathsf{poly}(\lambda)$*, there is no PPT adversary* $\mathcal{A}$ *with more than negligible probability of outputting an index* $\mathsf{ind} \in [\ell]$*, a set of keys* $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$*, a subset* $U \subseteq V$ *with* $|U| \geq t$*, message lists* $(m_i)_{i\in[\ell]}, (T_i)_{i\in[\ell]}$ *and signatures* $(\sigma_i)_{i\in[\ell]}$*, such that* σ_{ind} *is a valid signature on* T_{ind} *under a threshold number of keys, but does not facilitate correct decryption of* m_i*. More formally:*

$$\mathsf{AggVrfy}(\sigma_{\mathsf{ind}}, U, T_{\mathsf{ind}}) = 1, \text{ but}$$
$$\mathsf{Dec}(\mathsf{Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]}), (\sigma_i)_{i\in[\ell]}, U, V)_{\mathsf{ind}} \neq m_{\mathsf{ind}}.$$

Definition 69 (IND-CPA Security)**.** *A t-out-of-n SWE scheme* $\mathsf{SWE}' = (\mathsf{Enc}, \mathsf{Dec})$ *for a multi-signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Agg}, \mathsf{AggVrfy}, \mathsf{Prove}, \mathsf{Valid})$ *is* secure *if for all* $\lambda \in \mathbb{N}$*, such that* $t = \mathsf{poly}(\lambda)$*, and all* $\ell = \mathsf{poly}(\lambda)$*, subsets* $SC \subseteq [\ell]$ *(signifying the indices for which challenge messages are inserted), there is no PPT adversary* $\mathcal{A}$ *that has more than negligible advantage in the experiment* $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$ *as defined below:*

1. *Let* H_{pr} *be a fresh hash function from a keyed family of hash functions, available to the experiment and* $\mathcal{A}$*.*
2. *The experiment generates* $n - t + 1$ *key pairs for* $i \in \{t, \ldots, n\}$ *as* $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ *and provides* vk_i *as well as* $\mathsf{Sig.Prove}^{H_{pr}}(\mathsf{vk}_i, \mathsf{sk}_i)$ *for* $i \in \{t, \ldots, n\}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *inputs* $VC = (\mathsf{vk}_1, \ldots, \mathsf{vk}_{t-1})$ *and* $(\pi_1, \ldots, \pi_{t-1})$*. If for any* $i \in [t-1]$*,* $\mathsf{Sig.Valid}(\mathsf{vk}_i, \pi_i) = 0$*, we abort. Else, we define* $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$*.*
4. $\mathcal{A}$ *gets to make signing queries for pairs* (i, T)*. If* $i < t$*, the experiment aborts, else it returns* $\mathsf{Sig.Sign}(\mathsf{sk}_i, T)$*.*
5. *The adversary announces challenge messages* m_i^0, m_i^1 *for* $i \in SC$*, a list of messages* $(m_i)_{i\in[\ell]\setminus SC}$ *and a list of signing reference messages* $(T_i)_{i\in[\ell]}$*. If a signature for a* T_i *with* $i \in SC$ *was previously queried, we abort.*
6. *The experiment flips a bit* $b \leftarrow\!\!\$\ \{0, 1\}$*, sets* $m_i = m_i^b$ *for* $i \in SC$ *and sends* $\mathsf{Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]})$ *to* $\mathcal{A}$*.*
7. $\mathcal{A}$ *gets to make further signing queries for pairs* (i, T)*. If* $i \geq t$ *and* $T \neq T_i$ *for all* $i \in SC$*, the experiment returns* $\mathsf{Sig.Sign}(\mathsf{sk}_i, T)$*, else it aborts.*
8. *Finally,* $\mathcal{A}$ *outputs a guess* b'*.*
9. *If* $b = b'$*, the experiment outputs 1, else 0.*

We define $\mathcal{A}$*'s advantage by* $\mathsf{Adv}_{\mathsf{Sec}}^{\mathcal{A}} = |\Pr\left[\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda) = 1\right] - \frac{1}{2}|$.

Remark on Adaptivity. We note that we fix the positions of the corrupted keys to positions 1 to $t - 1$ for notational convenience. Our SWE construction in this

chapter does not depend on the ordering of the keys in a crucial way and achieves adaptivity in the sense of the adversary choosing corrupted keys and inserting them at arbitrary indices into VC. This is in contrast to our definition of compact SWE Def. 78, where prior knowledge of the indices of corrupted keys is crucial.

We now introduce a verifiable variant of our SWE scheme. The goal is to ensure that ciphertexts are provably consistent with a tuple of plaintext messages $(m_i)_i$, for which the prover demonstrates that they satisfy an arbitrary NP relation, enabling expressive proofs about encrypted data.

In our protocol, efficient decryption will rely on each message m_i lying in the message space $\{0,\dots,2^k-1\}$. Without verifiability, a receiver of an SWE ciphertext has no guarantee that it can be decrypted efficiently before getting the relevant signatures to actually decrypt, as the plaintext message might be out of these bounds. The ability to show more complex properties of the underlying message can be crucial when SWE is used in a larger system, where the receiver's actions might be based on trust that they can later obtain a message of a specific shape, e.g., a signed transaction.

Definition 70 (Verifiable Signature-Based Witness Encryption)**.** *A scheme* $\mathsf{SWE}' = (\mathsf{Enc},\mathsf{Dec},\mathsf{Prove},\mathsf{Vrfy})$ *is a verifiable SWE for relation* $\mathcal{R}$*, if* $\mathsf{Enc},\mathsf{Dec}$ *are as above and* $\mathsf{Prove},\mathsf{Vrfy}$ *are a NIZK proof system for a language given by the following induced relation* $\mathcal{R}'$*, where* $V = (\mathsf{vk}_1,\dots,\mathsf{vk}_n)$ *is a set of keys:*

$$
\begin{aligned}
&(V,(T_i)_{i\in[\ell]},\mathsf{ct}),((m_i)_{i\in[\ell]},w,r)) \in \mathcal{R}' \Leftrightarrow \\
&\qquad \mathsf{ct} = \mathsf{Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]}); r) \text{ and } \forall i \in [\ell] : m_i \in \{0,\dots,2^k-1\} \\
&\qquad \text{and } (m = \sum_{i\in[\ell]} 2^{(i-1)k} m_i, w) \in \mathcal{R}
\end{aligned}
$$

4.4 BLS Signatures with Modified Aggregation

In this section we describe a modified aggregate BLS signature scheme. We show our signature to provide secure aggregation of signatures across multiple messages that may contain duplicates, making them aggregate as well as multi-signatures. Looking ahead, we will instantiate SWE in § 4.5 for the multi-signature version of this signature scheme. The key generation, signing and verification algorithms will be identical to regular BLS signatures [BLS01], but we add Lagrange exponents to the signature aggregation process compared to standard aggregate BLS signatures [BGLS03].

4.4.1 Description

Our system parameters include two base groups $\mathbb{G}_1,\mathbb{G}_2$ of prime order p with generators g_1,g_2 which have a bilinear map $e:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$ into a target group $\mathbb{G}_T$ with generator g. Also we assume full-domain hash functions $H:\{0,1\}^*\to\mathbb{G}_1$ and $H_2, H_{pr}:\{0,1\}^*\to\mathbb{Z}_p$. Let $(\mathsf{Schnorr.Prove},\mathsf{Schnorr.Valid})$ be the non-interactive variant of the well-known Schnorr proofs due to Fischlin [Fis05] discussed in § 2.11.5.1.

Construction 4. *The algorithms of our aggregate signature scheme* $\mathsf{Sig}_{\mathsf{agg}} = (\mathsf{KeyGen},\mathsf{Sign},\mathsf{Vrfy},\mathsf{Agg},\mathsf{AggVrfy},\mathsf{Prove},\mathsf{Valid})$ *are given as follows.*

$\mathsf{Sig}_{\mathsf{agg}}.\mathsf{KeyGen}(1^\lambda)$:

- *Randomly pick* $x \leftarrow\!\!\$\ \mathbb{Z}_p$.
- *Output* $(\mathsf{vk} = {g_2}^x, \mathsf{sk} = x)$.

$\mathsf{Sig}_{\mathsf{agg}}.\mathsf{Sign}(\mathsf{sk}, T)$:

- *Output* $H(T)^{\mathsf{sk}}$.

$\mathsf{Sig}_{\mathsf{agg}}.\mathsf{Vrfy}(\mathsf{vk}, T, \sigma)$:

- *Compute* $h = H(T)$.
- *If* $(e(\sigma, g_2) = e(h, \mathsf{vk}))$, *output 1, else output 0.*

$\mathsf{Sig}_{\mathsf{agg}}.\mathsf{Agg}((\sigma_1, \ldots, \sigma_k), (\mathsf{vk}_1, \ldots, \mathsf{vk}_k))$:

- *Compute* $\xi_i = H_2(\mathsf{vk}_i)$ *for* $i \in [k]$.
- *Compute* $L_i = \prod_{j \in [k], i \neq j} \frac{-\xi_j}{\xi_i - \xi_j}$ *for* $i \in [k]$.
- *Output* $\sigma \leftarrow \prod_{i \in [k]} \sigma_i^{L_i}$.

$\mathsf{Sig}_{\mathsf{agg}}.\mathsf{AggVrfy}(\sigma, (\mathsf{vk}_1, \ldots, \mathsf{vk}_k), (T_1, \ldots, T_k))$:

- *If* $e(\sigma, g_2) = \prod_{i \in [k]} e(H(T_i), \mathsf{vk}_i)^{L_i}$, *output 1. Output 0 otherwise.*

$\mathsf{Sig}_{\mathsf{agg}}.\mathsf{Prove}(\mathsf{vk}, \mathsf{sk})$:

- *Output* $\mathsf{Schnorr.Prove}^{H_{pr}}(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{Sig}_{\mathsf{agg}}.\mathsf{Valid}(\mathsf{vk}, \pi)$:

- *Output* $\mathsf{Schnorr.Valid}^{H_{pr}}(\mathsf{vk}, \pi)$.

(Schnorr.Prove, Schnorr.Valid) fulfill the requirements of an online-extractable zero-knowledge proof of knowledge for the relation $\mathcal{K} = \{(g_2^x, x) : x \in \mathbb{Z}_p\}$, as is shown in [Fis05]. We use the Fischlin transform, rather than the Fiat-Shamir transform [FS87], due to its online extractability.

4.4.2 Proofs

We will now show that $\mathsf{Sig}_{\mathsf{agg}}$ fulfills the requirements for aggregate signatures stated in Def. 63. We have discussed that $(\mathsf{Sig}_{\mathsf{agg}}.\mathsf{Prove}, \mathsf{Sig}_{\mathsf{agg}}.\mathsf{Valid})$ constitutes a valid online-extractable proof of knowledge for the key relation already.

Theorem 10. $\mathsf{Sig}_{\mathsf{agg}}$ *is correct.*

Proof. The first part of correctness follows directly from [BLS01] as the algorithms and definitions are identical to standard BLS.

Let $\lambda \in \mathbb{N}$, $k = \mathsf{poly}(\lambda)$, messages $T_1, \ldots, T_k$, a set of public keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_k)$ and signatures $\sigma_1, \ldots, \sigma_k$ be given, such that $\mathsf{Vrfy}(\mathsf{vk}_i, T_i, \sigma_i) = 1$ for $i \in [k]$. This means that $e(\sigma_i, g_2) = e(H(T_i), \mathsf{vk}_i)$ for $i \in [k]$. Let $\sigma \leftarrow \mathsf{Agg}((\sigma_1, \ldots, \sigma_k), V)$. We need to show $\mathsf{AggVrfy}(\sigma, V, (T_1, \ldots, T_k)) = 1$.

By construction, the aggregate signature is $\sigma = \prod_{i \in [k]} \sigma_i^{L_i}$ for L_i as defined in our algorithm. Now, in our call to AggVrfy, it holds

$$e(\sigma, g_2) = e(\prod_{i \in [k]} \sigma_i^{L_i}, g_2) = \prod_{i \in [k]} e(\sigma_i, g_2)^{L_i} = \prod_{i \in [k]} e(H(T_i), \mathsf{vk}_i)^{L_i}.$$

Therefore, the output is 1. □

Theorem 11. *Assume that H is modelled as a random oracle.* $\mathsf{Sig}_{\mathsf{agg}}$ *is unforgeable, given that the computational Co-Diffie-Hellman assumption holds for* $(\mathbb{G}_1, \mathbb{G}_2)$.

Proof. Our proof takes some ideas from [RY07] and [BGLS03], but mainly works due to the full simulation extractability of Schnorr. We will regard adversaries $\mathcal{A}$ that are permitted to issue at most a polynomial number q_S of signature queries.

We assume that $\mathcal{A}$ has non-negligible winning probability ε and will only output a forgery $(T_1, \ldots, T_k), (\mathsf{vk}_2, \ldots, \mathsf{vk}_k), (\pi_2, \ldots, \pi_k), \sigma^*$ where $\mathsf{Valid}(\mathsf{vk}_i, \pi_i) = 1$ for all

$i \in \{2, \ldots, k\}$ and T_1 was not queried to the signing oracle, otherwise they would not be able to win in the unforgeability experiment. Assuming control over the random oracle, we can use the fact that $(\mathsf{Prove}, \mathsf{Valid})$ constitutes an online-extractable proof of knowledge.

We define a reduction against co-CDH:

The challenger receives a tuple $g_1, h_1 = g_1^x, g_2, h_2 = g_2^x, h$ with the public parameters g_1, g_2 as generators and is required to output h^x.

Let S be the simulator for the Schnorr proof system due to the full simulation extractability (Def. 40). The experiment executes $(H_{pr}, \tau_0) \leftarrow S(0, 1^\lambda)$ and provides oracle access to H_{pr} to $\mathcal{A}$.

The adversary gets access to the publicly known $\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, H_2$. Queries to the hash function H are being programmed by the reduction.

The reduction provides as public key $\mathsf{vk}^* = h_2$ to $\mathcal{A}$.

Any queries to H are answered as follows:

- If the message T was previously queried, we respond as before.
- Else, with probability δ we select its hash as $h \cdot g_1^\theta$ for random $\theta \leftarrow\!\!\$\ [p]$ and save T to a special list C.
- Otherwise, we select its hash as g_1^θ for random $\theta \leftarrow\!\!\$\ [p]$
- In both cases, we save $A[T] = \theta$

If $\mathcal{A}$ queries for a proof of knowledge for vk^* we call $(H_1, \pi^*, \tau') \leftarrow S(1, \mathsf{vk}^*, \tau, \texttt{YES})$, respond with π^* and replace the oracle H_{pr} by H_1 in future calls. By zero-knowledge of Schnorr, this is indistinguishable from the real experiment's output except with negligible probability. We note, that all $\mathsf{vk} \in \mathbb{G}_2$ actually have a valid secret key, making the `YES` call justified.

Any queries to the signing oracle for a message T under vk^* are answered as follows:

- We determine $H(T)$.
- If T is in the special list C, we abort.
- Otherwise, we know the hash $H(T) = g_1^\theta$.
- We output as signature h_1^θ. Since $h_1 = g_1^x$ for some x such that $\mathsf{vk} = h_2 = g_2^x$, this is simply $g_1^{\theta x} = H(T)^x$, which is a valid signature under vk.

Once we receive $(T_1, \ldots, T_k), (\mathsf{vk}_2, \ldots, \mathsf{vk}_k), (\pi_2, \ldots, \pi_k), \sigma^*$ from $\mathcal{A}$,

- For $i \in \{2, \ldots, k\}$, we call the PPT extractor $\mathcal{E}(\mathsf{vk}_i, \pi_i, Q_\mathcal{A})$ where $Q_\mathcal{A}$ are the queries to H_{pr} so far.
- Since $\mathsf{Valid}(\mathsf{vk}_i, \pi_i)$ holds by assumption, and due to full simulation extractability of Schnorr, we can extract the sk_i except with negligible probability and save them to a table $P[\mathsf{vk}_i] = \mathsf{sk}_i$. If we fail to extract for any index, we abort.
- We check whether T_1 is on the list C and whether σ^* is a valid forgery. If this is not the case, we abort.
- If any of the keys $\mathsf{vk}_2, \ldots, \mathsf{vk}_n$ is equal to vk^*, the adversary may not have asked for our simulated proof on vk^* and therefore must have given a proof of their own which we extracted from. Thus, we have $x = P[\mathsf{vk}_i]$ for that key by definition and thus can output h^x directly.
- Otherwise, it holds $e(\sigma^*, g_2) = \prod_{i \in [k]} e(H(T_i), \mathsf{vk}_i)^{L_i}$ where we consider $\mathsf{vk}^* = \mathsf{vk}_1$.
- Now for $i \in \{2, \ldots, k\}$, we can make partial signatures $\sigma_i = H(T_i)^{P[\mathsf{vk}_i] L_i}$ with $e(\sigma_i, g_2) = e(H(T_i), \mathsf{vk}_i)^{L_i}$.

- We set σ' to be $\sigma^* / \prod_{i\in\{2,\dots,k\}}(\sigma_i)$. Now, it clearly holds $e(\sigma', g_2) = e(H(T_1), \mathsf{vk}_1)^{L_1}$. Therefore, we have that $\sigma' = (h \cdot g_1^{A[T_1]})^{xL_1}$ and we output $\left(\frac{\sigma'}{h_1^{A[T_1]L_1}}\right)^{-L_1} = h^x$. This holds as $h_1 = g_1^x$.

Clearly, if no abort occurs, the output is indeed h^x.

Now, what is the success probability? Assume the adversary $\mathcal{A}$ has advantage ε in winning the unforgeability experiment. If they win, they can either query us for a proof on vk^* or be able to include vk^* in their forgery.

$\mathcal{A}$ can only succeed, if its combined probability of successfully registering all keys $\mathsf{vk}_2, \dots, \mathsf{vk}_k$ it chooses is at least ε, making every one of these probabilities non-negligible. By full simulation extractability of our proof of knowledge and a union bound, this means we can extract all secret keys in polynomial time except with negligible probability. We note that since the hashes and vk^* are distributed uniformly at random, this looks indistinguishable from the real experiment for $\mathcal{A}$ unless they request a signature for one of the messages where T is in C. This probability can be bounded by $(1-\delta)^{q_S}$, assuming $\mathcal{A}$ only queries for messages once, as the probability is clearly independent for every message requested. Conditioned on no such request being made, we have a probability of ε of the adversary winning. Since the hash(es) which we created as $h \cdot g_1^\theta$ are i.i.d. in the view of $\mathcal{A}$, we then have a probability of δ of the first message T_1 in fact being such that we don't abort.

This gives us a winning probability negligibly worse than $(1-\delta)^{q_S} \cdot \delta \cdot \varepsilon$. By appropriately choosing $\delta = 1/q_S$ we get $(1-1/q_S)^{q_S} \cdot 1/q_S \cdot \varepsilon \geq 0.1/q_S \cdot \varepsilon$, assuming that $q_S \geq 2$, as $(1-1/x)^x$ converges to $1/e$ in a strictly increasing manner. Therefore the advantage of the reduction will be non-negligible, if ε is not negligible.

The reduction is clearly running in polynomial time if $\mathcal{A}$ is.

□

4.5 Construction of Signature-Based Witness Encryption

In this section we provide an instantiation for a t-out-of-n SWE scheme that is compatible with the modified BLS signature scheme defined in the previous section. Its security is based on the bilinear Diffie-Hellman assumption. Our construction is specifically optimized to push as many operations as possible into the source group $\mathbb{G}_2$. This leads to significant performance improvements over a naive approach if we choose $\mathbb{G}_2$ to be the one of the two source groups for which group operations are cheaper.

4.5.1 Construction

We describe two variants of the construction. The *base construction* SWE' assumes that the input messages T_i are all distinct. While this variant achieves slightly more compact ciphertexts than our other variant, the distinctness requirement can be restrictive in practice. To address this limitation, we present an *enhanced construction* SWE that removes the need for message uniqueness by introducing additional randomness. This enhanced variant incurs a minor overhead but should usually be preferred in practice due to its broader applicability. Accordingly, it will be our main focus, while including the base construction for completeness, and to offer an alternative in settings where its particular trade-offs may be desirable.

We restrict the message space to k-bit messages, i.e., $m_i \in \{0, \dots, 2^k - 1\}$ for all $i \in [\ell]$ and a parameter k. The choice of k will inform a trade-off between how

many bits we can pack into a ciphertext and the space and time required to implement decryption. Specifically, when decrypting, we need to extract m_i from $g_T^{m_i}$ by computing a dlog in $\mathbb{G}_T$.

Further, we assume all algorithms in this section to have oracle access to full-domain hash functions $H : \{0,1\}^* \to \mathbb{G}_1$ and $H_2, H_{pr} : \{0,1\}^* \to \mathbb{Z}_p$ as used in our signature scheme, see § 4.4.1.

Enhanced Construction. Let us start with our enhanced construction:

Construction 5. *The message space is restricted to* $(m_i)_{i\in[\ell]} \in \{0,\dots,2^k-1\}^\ell$. *The algorithms* $(\mathsf{SWE.Enc}, \mathsf{SWE.Dec})$ *are given as follows:*

$\mathsf{SWE.Enc}(1^\lambda, (\mathsf{vk}_j)_{j\in[n]}, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]})$:

- *Choose random* $r, r_j \leftarrow\!\!\$\ \mathbb{Z}_p$ *for* $j \in \{0,\dots,t-1\}$.
- *Let* $f(x) = \sum_{j=0}^{t-1} r_j \cdot x^j$. *This will satisfy* $f(0) = r_0$.
- *For* $j \in [n]$, *set* $\xi_j = H_2(\mathsf{vk}_j)$, $s_j = f(\xi_j)$.
- *For* $i \in [\ell]$ *choose random* $\alpha_i \leftarrow\!\!\$\ \mathbb{Z}_p$.
- *Compute* $c = g_2^r$, $a_i = c^{\alpha_i}$, $t_i = H(T_i)^{\alpha_i}$ *for* $i \in [\ell]$.
- *Choose* $h \leftarrow \mathbb{G}_2$ *uniformly at random.*
- *Compute* $c_0 = h^r \cdot g_2^{r_0}$.
- *For* $j \in [n]$, *compute* $c_j = \mathsf{vk}_j^r \cdot g_2^{s_j}$.
- *For* $i \in [\ell]$, *set* $c_i' = e(t_i, g_2^{r_0}) \cdot g_T^{m_i}$.
- *Output* $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c_i', a_i, t_i)_{i\in[\ell]})$.

$\mathsf{SWE.Dec}(\mathsf{ct}, (\sigma_i)_{i\in[\ell]}, U, V)$:

- *Parse* $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c_i', a_i, t_i)_{i\in[\ell]})$.
- *Parse* $V = (\mathsf{vk}_1, \dots, \mathsf{vk}_n)$, $U = (\mathsf{vk}_1', \dots, \mathsf{vk}_k')$.
- *If* $k < t$ *or* $U \not\subseteq V$, *abort.*
- *Define as* I *the indices* $j \in [n]$ *s.t.* $\mathsf{vk}_j \in U$.
- *Compute* $\xi_j = H_2(\mathsf{vk}_j)$ *for* $j \in I$.
- *Compute* $L_j = \prod_{i\in I, i\neq j} \frac{-\xi_i}{\xi_j - \xi_i}$ *for* $j \in I$.
- *Compute* $c^* = \prod_{j\in I} c_j^{L_j}$.
- *For* $i \in [\ell]$, *compute* $z_i = c_i' \cdot e(\sigma_i, a_i)/e(t_i, c^*)$.
- *For* $i \in [\ell]$, *compute* $m_i' = \mathsf{dlog}_{g_T}(z_i)$.
- *Output* $(m_i')_i$.

We draw the reader's attention to efficiency considerations and design choices:

- We chose our parameter k depending on practical considerations, such that the extraction of the discrete logarithm in $\mathsf{SWE.Dec}$ does not cause a large overhead. We only decrypt values from a bounded message space and can therefore precompute a lookup table using the baby-step giant-step method [Sha71]. This yields efficient decryption with time complexity $\mathcal{O}(2^{k/2})$, where k is the bit-length of the message. The precomputation requires storing a table of size $\mathcal{O}(2^{k/2})$ group elements, which can be reused across decryptions.
- Another potential bottleneck are the group multiplications necessary to compute c^*. Notice here, that we only do this potentially expensive computation once and reuse it for all message parts. This is on the condition that in the use of our protocol, the sets of signers are the same for all T_i. If they aren't or only some of the multi-signatures σ_i are given, it is still possible to compute the m_i for which we have signatures on T_i, but we may have to compute c^* for all relevant sets U of signers.

- We note that h and c_0 as computed by the encryption are redundant terms, that are not used in decryption. Looking ahead, these terms are included to make our ciphertexts into a homomorphic commitment to the underlying plaintext (see § 4.6.2). Further, we choose m_i from $\mathbb{Z}_p$ to enable the usage of efficient bulletproofs to go with these commitments. This will help us in adding a proof layer $\mathsf{SWE.Prove}, \mathsf{SWE.Vrfy}$ that makes SWE verifiable.

Base Construction. In the base construction, we do not re-randomize the c'_i with fresh α_i for each slot $i \in [\ell]$, this can be seen as SWE with setting all $\alpha_i = 1$. This construction remains secure when all T_i are distinct.

Construction 6. *The message space is restricted to* $(m_i)_{i\in[\ell]} \in \{0,\dots,2^k-1\}^\ell$. *The algorithms* $(\mathsf{SWE'.Enc}, \mathsf{SWE'.Dec})$ *of our base construction are given as follows:*

$\mathsf{SWE'.Enc}(1^\lambda, (\mathsf{vk}_j)_{j\in[n]}, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]})$:

- *Choose random* $r, r_j \leftarrow_\$ \mathbb{Z}_p$ *for* $j \in \{0,\dots,t-1\}$.
- *Let* $f(x) = \sum_{j=0}^{t-1} r_j \cdot x^j$. *This will satisfy* $f(0) = r_0$.
- *For* $j \in [n]$, *set* $s_j = f(\xi_j)$, *where* $\xi_j = H_2(\mathsf{vk}_j)$.
- *Compute* $c = g_2^r$.
- *Choose* $h \leftarrow \mathbb{G}_2$ *uniformly at random.*
- *Compute* $c_0 = h^r \cdot g_2^{r_0}$.
- *For* $j \in [n]$, *compute* $c_j = \mathsf{vk}_j^r \cdot g_2^{s_j}$.
- *For* $i \in [\ell]$, *set* $c'_i = e(H(T_i), g_2^{r_0}) \cdot g_T^{m_i}$.
- *Output* $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c'_i)_{i\in[\ell]})$.

$\mathsf{SWE'.Dec}(\mathsf{ct}, (\sigma_i)_{i\in[\ell]}, U, V)$:

- *Parse* $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c'_i)_{i\in[\ell]})$.
- *Parse* $V = (\mathsf{vk}_1, \dots, \mathsf{vk}_n)$.
- *Parse* $U = (\mathsf{vk}'_1, \dots, \mathsf{vk}'_k)$. *If* $k < t$ *or* $U \not\subseteq V$ *abort.*
- *Define as* I *the indices* $j \in [n]$ *such that* $\mathsf{vk}_j \in U$.
- *Compute* $\xi_j = H_2(\mathsf{vk}_j)$ *for* $j \in I$.
- *Compute* $L_j = \prod_{i\in I, i\neq j} \frac{-\xi_i}{\xi_j - \xi_i}$ *for* $j \in I$.
- *Compute* $c^* = \prod_{j\in I} c_j^{L_j}$.
- *For* $i \in [\ell]$, *compute* $z_i = c'_i \cdot e(\sigma_i, c)/e(H(T_i), c^*)$.
- *For* $i \in [\ell]$, *compute* $m'_i = \mathsf{dlog}_{g_T}(z_i)$.
- *Output* $(m'_i)_i$.

4.5.2 Efficiency

We will briefly analyze the number of group operations in each group required for encryption and decryption in Table 4.1. We regard the number of n and ℓ to be fixed and give upper bounds on the operations needed. We require no multiplications or exponentiations in $\mathbb{G}_1$. In practice $\mathbb{G}_1$ and $\mathbb{G}_2$ can be reversed if $\mathbb{G}_1$ is more efficient in a given implementation.

The cost of allowing repeated reference messages T_i in SWE is 2ℓ additional exponentiations in the cheaper source group $\mathbb{G}_2$ and 2ℓ more ciphertext parts in the output. This is a relatively mild overhead.

For concrete efficiency considerations, a reference implementation of the base scheme $\mathsf{SWE'.Enc}, \mathsf{SWE'.Dec}$ was developed for the curve `BLS12-381`, which is used, for example, in Zcash and Ethereum 2.0. For a 381-bit message and a committee of size 500, the encryption time is 9.8s and decryption is 14.8s. The scheme remains practical for a committee of size 2000 with an encryption time of 58s and decryption

TABLE 4.1: Analysis of SWE' Efficiency in Group Operations

		SWE' encryption	SWE' decryption	SWE encryption	SWE decryption
evaluations of H, H_2		ℓ, n	ℓ, n	ℓ, n	$0, n$
Multiplications, Exponentiations	in $\mathbb{G}_1$	$0, 0$	$0, 0$	$0, \ell$	$0, 0$
	in $\mathbb{G}_2$	$n+1, 2n+3$	$n-1, n$	$n+1, 2\ell+2n+3$	$n-1, n$
	in $\mathbb{G}_T$	ℓ, ℓ	$2\ell, 0$	ℓ, ℓ	$2\ell, 0$
pairing evaluations		ℓ	2ℓ	ℓ	2ℓ
dlog in $\mathbb{G}_T$		0	ℓ	0	ℓ

time of 218s. From the experimental timings we chose $k = 24$, yielding a message space of $m_i \in \{0, \ldots, 2^{24} - 1\}$. We refer the reader to the full version of our paper [DHMW22] for implementation details and performance evaluations. We note that, by our analysis of operations in Table 4.1, $\mathsf{SWE.Dec}$ and $\mathsf{SWE.Enc}$ should only show mild overhead compared to $\mathsf{SWE}'.\mathsf{Dec}$ and $\mathsf{SWE}'.\mathsf{Enc}$, so we can expect SWE to still be very practical.

4.5.3 Proofs

We will now show that SWE fulfills the requirements for a signature-based witness encryption due to Def. 67 and SWE' does so, if the reference messages T_i involved are all distinct.

Theorem 12. SWE' *and* SWE *for the signature scheme* $\mathsf{Sig}_{\mathsf{agg}}$ *both have robust correctness, given that* H_2 *is collision resistant.*

Proof. Let $\lambda \in \mathbb{N}$, $\ell = \mathsf{poly}(\lambda)$ be given. Let us assume towards contradiction, that there is an adversary $\mathcal{A}$ with non-negligible winning probability against robust correctness as in Def. 68. That is, $\mathcal{A}$ provides with non-negligible probability an index $\mathsf{ind} \in [\ell]$, keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, a subset $U \subseteq V$ with $|U| \geq t$, reference messages $(T_i)_{i\in[\ell]}$, messages $(m_i)_{i\in[\ell]}$ and $(\sigma_i)_{i\in[\ell]}$ with $\mathsf{AggVrfy}(\sigma_{\mathsf{ind}}, U, T_{\mathsf{ind}}) = 1$, but $\mathsf{Dec}(\mathsf{Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]}), (\sigma_i)_{i\in[\ell]}, U, V)_{\mathsf{ind}} \neq m_{\mathsf{ind}}$

Let us consider a hybrid $\mathcal{H}_1$: It is identical to this robust correctness experiment, except if $H_2(\mathsf{vk}_i) = H_2(\mathsf{vk}_j)$ for any $i, j \in [n], i \neq j$, we abort. Clearly, except with negligible probability running $\mathcal{H}_1$ with $\mathcal{A}$ has the same outcome as the original experiment. Otherwise, we could build a reduction against the collision resistance of H_2.

We now show, that in $\mathcal{H}_1$, the probability of winning for the adversary is 0. Let any index $\mathsf{ind} \in [\ell]$, keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, a subset $U \subseteq V$ with $|U| \geq t$, reference messages $(T_i)_{i\in[\ell]}$, messages $(m_i)_{i\in[\ell]}$ and $(\sigma_i)_{i\in[\ell]}$ be given by $\mathcal{A}$. Let I be the set of all indices i for which $\mathsf{vk}_i \in U$.

We note, that since g_2 is a generator of $\mathbb{G}_2$, there exist x_i such that $\mathsf{vk}_i = g_2^{x_i}$ for $i \in [n]$. We assume $\mathsf{AggVrfy}(\sigma_{\mathsf{ind}}, U, T_{\mathsf{ind}}) = 1$, that is $e(\sigma_{\mathsf{ind}}, g_2) = \prod_{i\in I} e(H(T_{\mathsf{ind}}), \mathsf{vk}_i)^{L_i}$. Otherwise, $\mathcal{A}$ could not win. We now show that

$$\mathsf{Dec}(\mathsf{Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]}), (\sigma_i)_{i\in[\ell]}, U, V)_{\mathsf{ind}} = m_{\mathsf{ind}}$$

- For the base construction SWE',
  the ciphertext $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c'_i)_{i\in[\ell]}) = \mathsf{Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]})$ has the relevant components for decryption $c = g_2^r$, $c_j = \mathsf{vk}_j^r \cdot g_2^{s_j}$ for $j \in [n]$ and $c'_i = e(H(T_i), g_2)^{r_0} \cdot g^{m_i}$ for $i \in [\ell]$, where $s_j = f(\xi_j)$ for a polynomial such that $f(0) = r_0$.

- For the enhanced construction SWE, additionally, $a_i = c^{\alpha_i}, t_i = H(T_i)^{\alpha_i}$ are given and we set $c'_i = e(H(T_i), g_2)^{r_0 \alpha_i} \cdot g^{m_i}$

Now, in both cases the ξ_j, L_j computed by Dec are identical to those used in Enc and in $\mathsf{Sig_{agg}.Sign}$. Note that since no two distinct $\mathsf{vk}_j \neq \mathsf{vk}_{j'}$ collide under H_2, the support points $\xi_j = H_2(\mathsf{vk}_j)$ are all distinct and $|I| \geq t$ so Lagrange interpolation will correctly recover r_0 from the s_j by computing $r_0 = f(0) = \sum_{j \in I} s_j L_j$. Thus it holds that

$$c^* = \prod_{j \in I} c_j^{L_j} = \left(\prod_{j \in I} \mathsf{vk}_j^{L_j} \right)^r \cdot \prod_{j \in I} g_2^{s_j L_j} = (\mathsf{vk}^*)^r \cdot g_2^{r_0}.$$

where $\mathsf{vk}^* := \prod_{j \in I} \mathsf{vk}_j^{L_j} = g_2^{\sum_{j \in I} x_j L_j}$.

- Now, in the case of the base SWE', it holds for index ind:

$$\begin{aligned} e(H(T_{\mathsf{ind}}), c^*) &= e(H(T_{\mathsf{ind}}), g_2^{r \cdot \sum_{j \in I} x_j L_j} \cdot g_2^{r_0}) \\ &= \prod_{j \in I} e(H(T_{\mathsf{ind}}), \mathsf{vk}_j)^{r \cdot L_j} \cdot e(H(T_{\mathsf{ind}}), g_2)^{r_0} \\ &= e(\sigma_{\mathsf{ind}}, g_2)^r \cdot e(H(T_{\mathsf{ind}}), g_2)^{r_0} \\ &= e(\sigma_{\mathsf{ind}}, c) \cdot e(H(T_{\mathsf{ind}}), g_2)^{r_0}. \end{aligned}$$

  Since $c'_{\mathsf{ind}} = e(H(T_{\mathsf{ind}}), g_2)^{r_0} \cdot g_T^{m_{\mathsf{ind}}}$, it follows that $z_{\mathsf{ind}} = c'_{\mathsf{ind}} \cdot e(\sigma_{\mathsf{ind}}, c) / e(H(T_{\mathsf{ind}}), c^*) = g_T^{m_{\mathsf{ind}}}$. It follows for the ind-th output: $m'_{\mathsf{ind}} = \mathsf{dlog}_{g_T}(z_{\mathsf{ind}}) = m_{\mathsf{ind}}$, and robust correctness follows.
- In the case of the enhanced SWE, analogously it holds for index ind:

$$\begin{aligned} e(t_{\mathsf{ind}}, c^*) &= e(H(T_{\mathsf{ind}}), c^*)^{\alpha_{\mathsf{ind}}} = e(\sigma_{\mathsf{ind}}, c)^{\alpha_{\mathsf{ind}}} \cdot e(H(T_{\mathsf{ind}}), g_2)^{r_0 \alpha_{\mathsf{ind}}} \\ &= e(\sigma_{\mathsf{ind}}, a_{\mathsf{ind}}) \cdot e(t_{\mathsf{ind}}, g_2)^{r_0} \end{aligned}$$

  and thus decryption via $z_{\mathsf{ind}} = c'_{\mathsf{ind}} \cdot e(\sigma_{\mathsf{ind}}, a_{\mathsf{ind}}) / e(t_{\mathsf{ind}}, c^*) = g_T^{m_{\mathsf{ind}}}$ also yields robust correctness.

□

Theorem 13. *Assume that the hash functions H, H_2, H_{pr} are modelled as random oracles and all references T_i are distinct. Then SWE' for the signature scheme $\mathsf{Sig_{agg}}$ is secure under the BDH assumption in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. The security reduction is tight.*

Proof. We actually show a little bit more, namely, that the ciphertexts are essentially pseudorandom at the indices where the challenge messages are included.

Lemma 4. *Specifically, assuming that the hash functions H, H_2, H_{pr} are modelled as random oracles and all references T_i are distinct and under the BDH assumption in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, the experiment $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$ is indistinguishable from a modified experiment $\mathsf{Exp}'_{\mathsf{Sec}}$, in which, instead of a well-formed $\mathsf{ct} = (h, c, c_0, (c_j)_{j \in [n]}, (c'_i)_{i \in [\ell]}) = \mathsf{Enc}(1^\lambda, V, (T_i)_{i \in [\ell]}, (m_i)_{i \in [\ell]})$, we replace (c'_i) by a random element u_i from $\mathbb{G}_T$ for all challenge indices $i \in SC$.*

As the c'_i are the only parts of ct that depend on m_i^b, this implies security.

Assume that $\mathcal{A}$ is a PPT adversary distinguishing $\mathsf{Exp}'_{\mathsf{Sec}}$ from the real $\mathsf{Exp}_{\mathsf{Sec}}$ with advantage ε. We will construct a PPT distinguisher $\mathcal{D}$ with advantage ε against

the BDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. The complexity of $\mathcal{D}$ is essentially the same as that of $\mathcal{A}$.

Let $I = \{1, \ldots, t-1\}$ be the set of indices of verification keys vk_i which are chosen by the adversary. For each $i \in I$ let $\mathsf{vk}_i = g_2^{x_i}$ where $\mathsf{sk}_i = x_i$. In the following we will assume that the distinguisher $\mathcal{D}$ has access to all the signing keys x_i for $i \in I$. This can be achieved, by using the fact that $\mathsf{Sig}_{\mathsf{agg}}$ has an online-extractable proof of knowledge.

Let SC be the set of indices at which the challenge messages are included in encryption.

The distinguisher $\mathcal{D}$ receives as input a tuple $(g_1, h_1, v, g_2, h_2, w, z)$, where g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, $h_1 = g_1^x$ for some $x \in \mathbb{Z}_p$, $h_2 = g_2^x$ for the same $x \in \mathbb{Z}_p$, $v = g_1^\alpha$ for an $\alpha \in \mathbb{Z}_p$ and $w = g_2^r$ for an $r \in \mathbb{Z}_p$. The term $z \in \mathbb{G}_T$ is either of the form $g_t^{\alpha x r}$, in which case we say that this is a BDH tuple, or z is chosen uniformly random from $\mathbb{G}_T$, in which case we say this is a random tuple.

The distinguisher $\mathcal{D}$ simulates the security experiment for SWE', except in the way the verification keys vk_i for $i \in \bar{I} = \{t, \ldots, n\}$ are chosen, the corresponding signatures for these keys are computed and the challenge-ciphertext ct^* is computed.

It uses the simulator S due to the full simulation extractability of the $\mathsf{Schnorr}$ proof system to create H_{pr} and the outputs of Prove on its verification keys and it uses the extractor to gain access to the secret keys of the adversary.

To simulate the random oracles H, H_2 lazily, $\mathcal{D}$ initializes two lists $\mathcal{L}, \mathcal{L}_2 = \emptyset$. $\mathcal{D}$ first computes the following auxiliary terms.

- For $i \in [n]$ randomly draw $\xi_i \leftarrow\!\!\$\ \mathbb{Z}_p$.
- Define the Lagrange polynomials

$$L'_0(x) = \prod_{j \in I} \frac{x - \xi_j}{-\xi_j} \text{ and } L'_i(x) = \frac{x}{\xi_i} \prod_{j \in I \setminus \{i\}} \frac{x - \xi_j}{\xi_i - \xi_j}$$

  for $i \in I$. That is, the L'_i are an interpolation basis for the support points $\{0\} \cup \{\xi_j \mid j \in I\}$.
- For every $i \in \bar{I}$ choose $x'_i \leftarrow \mathbb{Z}_p$ uniformly at random and set $h_{1,i} = h_1^{L'_0(\xi_i)} \cdot g_1^{x'_i}$, $h_{2,i} = h_2^{L'_0(\xi_i)} \cdot g_2^{x'_i}$.
- Choose $y \leftarrow \mathbb{Z}_p$ uniformly at random and set $A = e(v, g_2)^y / z$ and $B = g_T^y / e(h_1, w)$.
- The reduction sets $(H_0, \tau_0) \leftarrow S(0, 1^\lambda)$. It also sets a counter $ctr = 1$.

The verification keys vk_i, random oracle queries, signature queries and the challenge ciphertext ct^* are now computed as follows.

- For every honest party with index $i \in \bar{I}$, $\mathcal{D}$ computes the verification key vk_i as $\mathsf{vk}_i = h_{2,i}$ and generates the associated proof of knowledge by $(H_{ctr}, \pi_i, \tau_{ctr}) \leftarrow S(ctr, \mathsf{vk}_i, \tau_{ctr-1}, \mathtt{YES})$, incrementing ctr after each call. Finally, we set $H_{pr} = H_{ctr}$ and make it available to $\mathcal{A}$. Due to the zero-knowledge property of the proof of knowledge, this is possible without noticeably changing the distribution of the output from either the real experiment $\mathsf{Exp}_{\mathsf{Sec}}$ or $\mathsf{Exp}'_{\mathsf{Sec}}$, except with negligible probability.
- When $\mathcal{A}$ sends (vk_i, π_i) for $i \in I$, we proceed as follows.
  - If $\mathsf{Valid}^{H_{pr}}(\mathsf{vk}_i, \pi_i) \neq 1$ for any $i \in I$ we return $\perp$.
  - Otherwise, we compute and store $\mathsf{sk}_i \leftarrow \mathcal{E}(\mathsf{vk}_i, \pi_i, Q_{\mathcal{A}})$, where $Q_{\mathcal{A}}$ denotes the queries to H_{pr} that $\mathcal{A}$ made so far.

  By full simulation extractability of $\mathsf{Schnorr}$, extraction of the secret keys succeeds except with negligible probability. Since $\mathcal{A}$ must give a valid proof for

all of its polynomially many keys, we extract all their secret keys except with negligible probability.

- Every query to H for a message $T \neq T_i$ for all $i \in SC$ (or before the challenge messages are announced) is answered as follows: If H has been queried on T before, retrieve the pair (T, α_T) from the list $\mathcal{L}$. Otherwise, choose α_T uniformly at random, add (T, α_T) to $\mathcal{L}$. Output $g_1^{\alpha_T}$. We will program H on $(T_i)_{i \in SC}$ specifically later on.
- Every query to H_2 on some input X is treated similarly: Initially, we add (vk_i, ξ_i) to $\mathcal{L}_2$ for every $i \in [n]$. If $\mathcal{L}_2$ has an entry for X, retrieve the pair (X, β_X) from the list $\mathcal{L}_2$. Otherwise, choose β_X uniformly at random, add (X, β_X) to $\mathcal{L}_2$. Output β_X.
- For every signature query of a message $T \neq T_i$ for all $i \in SC$ (or before the challenge messages are announced) for an honest party with index $i \in \bar{I}$, $\mathcal{D}$ computes the signature σ as follows. Determine $H(T)$ and retrieve the pair (T, α_T) from the list $\mathcal{L}$. Output $\sigma = h_{1,i}^{\alpha_T}$.
- $\mathcal{D}$ computes the challenge-ciphertext ct^* as follows.
  - Set $c = w$.
  - Draw randomly $\gamma \leftarrow \mathbb{Z}_p$.
  - Set $h = h_2 \cdot g_2^{\gamma}$.
  - Set $c_0 = w^{\gamma} \cdot g_2^{y}$.
  - For all $i \in I$ choose s_i uniformly at random and set $c_i = w^{x_i} \cdot g_2^{s_i}$.
  - For all $i \in \bar{I}$ we set $c_i = g_2^{L_0'(\xi_i)y + \sum_{j \in I} L_j'(\xi_i) \cdot s_j} \cdot w^{x_i'}$.
  - For all $j \in [\ell] \setminus SC$ set $c_j' = \frac{e(g_1, g_2)^{\alpha_{T_j} y}}{e(h_1, w)^{\alpha_{T_j}}} g_T^{m_j}$ where (T_j, α_{T_j}) is from $\mathcal{L}$.
  - For $i \in SC$ choose $\gamma_i, \delta_i \leftarrow \mathbb{Z}_p$ uniformly at random, program $H(T_i) = v^{\gamma_i} \cdot g_1^{\delta_i}$ and set $c_i' = A^{\gamma_i} \cdot B^{\delta_i} \cdot g_T^{m_i}$.

We will now show the following:

1. If $(g_1, h_1, v, g_2, h_2, w, z)$ follows the BDH distribution, i.e. $h_1 = g_1^x$, $h_2 = g_2^x$, $v = g_1^{\alpha}$, $w = g_2^r$ and $z = g_T^{\alpha x r}$, then $\mathcal{D}$ simulates the security experiment $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^{\lambda})$ of SWE' perfectly from the view of $\mathcal{A}$.
2. If $(g_1, h_1, v, g_2, h_2, w, z)$ follows the random distribution, i.e. $h_1 = g_1^x$, $h_2 = g_2^x$, $v = g_1^{\alpha}$, $w = g_2^r$ and $z = u$ for a uniformly random $u \leftarrow \mathbb{G}_T$, then $\mathcal{D}$ simulates the modified experiment $\mathsf{Exp}'_{\mathsf{Sec}}(\mathcal{A}, 1^{\lambda})$ of SWE' perfectly from the view of $\mathcal{A}$.

Thus, $\mathcal{A}$'s advantage in deciding between the two is at least ε and it will follow that the distinguishing advantage of $\mathcal{D}$ against BDH is at least ε.

We will first analyze the distribution of the vk_i, the signatures σ and the challenge-ciphertext components h, c_0, c and c_i.

We will first calculate the terms $h_{1,i}$ and $h_{2,i}$ for $i \in \bar{I}$. It holds that

$$h_{1,i} = h_1^{L_0'(\xi_i)} \cdot g_1^{x_i'} = g_1^{L_0'(\xi_i)x + x_i'} = g_1^{\tilde{x}_i}$$
$$h_{2,i} = h_2^{L_0'(\xi_i)} \cdot g_2^{x_i'} = g_2^{L_0'(\xi_i)x + x_i'} = g_2^{\tilde{x}_i},$$

where we set $\tilde{x}_i = L_0'(\xi_i)x + x_i'$. Note that since the x_i' are uniformly random, so are the $\tilde{x}_i$.

Hence, for the verification keys vk_i for $i \in \bar{I}$ it holds that

$$\mathsf{vk}_i = h_{2,i} = g_2^{\tilde{x}_i}.$$

Next, we consider the distribution of the signatures σ of a message T created upon a signing request for an honest key vk_i for $i \in \bar{I}$. It holds that

$$\sigma = h_{1,i}^{\alpha_T} = g_1^{\alpha_T \cdot \tilde{x}_i} = H(T)^{\tilde{x}_i}.$$

Regarding the challenge-ciphertext ct^*, let us make some definitions. We define $r_0 = y - rx$ and set f to be the (uniquely defined) polynomial of degree $t-1$ obtained by interpolating the pairs $(0, r_0), (\xi_i, s_i)_{i \in I}$. For $i \in \bar{I}$, we now set $s_i = f(\xi_i)$. Now, the following holds:

- $c = w = g_2^r$
- $h = h_2 \cdot g_2^{\gamma}$ is uniformly distributed
- $c_0 = w^{\gamma} \cdot g_2^y = g_2^{\gamma r} \cdot g_2^{y - xr + xr} = (g_2^{\gamma + x})^r \cdot g_2^{r_0} = h^r \cdot g_2^{r_0}$
- For $i \in I$ it holds that

$$c_i = w^{x_i} \cdot g_2^{s_i} = g_2^{r \cdot x_i} \cdot g_2^{s_i} = \mathsf{vk}_i^r \cdot g_2^{s_i}.$$

- For $i \in \bar{I}$ it holds that

$$\begin{aligned} c_i &= g_2^{L_0'(\xi_i) \cdot y + \sum_{j \in I} L_j'(\xi_i) s_j} \cdot w^{x_i'} \\ &= g_2^{L_0'(\xi_i) \cdot (rx + r_0) + r x_i' + \sum_{j \in I} L_j'(\xi_i) s_j} \\ &= g_2^{r(L_0'(\xi_i) x + x_i') + L_0'(\xi_i) r_0 + \sum_{j \in I} L_j'(\xi_i) s_j} \\ &= g_2^{r(L_0'(\xi_i) x + x_i') + f(\xi_i)} \\ &= g_2^{r \tilde{x}_i + s_i} \\ &= \mathsf{vk}_i^r \cdot g_2^{s_i}. \end{aligned}$$

Note that the s_i have the proper distribution: r_0 as well as the s_i for $i \in I$ are uniformly random and independent. Thus f is a uniformly random polynomial of degree $t-1$. Next, we consider the ciphertext components c_j' for $j \in [\ell] \setminus SC$. It holds

$$\begin{aligned} c_j' &= \frac{e(g_1, g_2)^{\alpha_{T_j} y}}{e(h_1, w)^{\alpha_{T_j}}} g_T^{m_j} \\ &= \frac{e(g_1, g_2)^{\alpha_{T_j} y}}{e(g_1^x, g_2^r)^{\alpha_{T_j}}} g_T^{m_j} \\ &= e(g_1, g_2)^{\alpha_{T_j}(y - xr)} g_T^{m_j} \\ &= e(g_1^{\alpha_{T_j}}, g_2)^{r_0} g_T^{m_j} \\ &= e(H(T_j), g_2)^{r_0} g_T^{m_j}. \end{aligned}$$

This conforms to the regular distribution.

We will finally consider the ciphertext components c_i' for $i \in SC$. In the first case, assume that $(g_1, h_1, v, g_2, h_2, w, z)$ follows the BDH distribution, i.e. $z = g_T^{\alpha x r}$. In this case, it holds that

$$A = e(v, g_2)^y / z = g_T^{\alpha(rx + r_0)} \cdot g_T^{-\alpha x r} = g_T^{\alpha r_0}$$

and

$$B = g_T^y / e(h_1, w) = g_T^{rx + r_0} \cdot g_T^{-xr} = g_T^{r_0}.$$

It follows that

$$H(T_i) = v^{\gamma_i} \cdot g_1^{\delta_i} = g_1^{\gamma_i \alpha + \delta_i} = g_1^{\alpha_i},$$

where we set $\alpha_i = \gamma_i \alpha + \delta_i$. Note that since δ_i is chosen uniformly random, α_i is distributed uniformly random.

Now, c_i' is distributed according to

$$\begin{aligned} c_i' &= A^{\gamma_i} \cdot B^{\delta_i} \cdot g_T^{m_i} \\ &= g_T^{\alpha r_0 \gamma_i} \cdot g_T^{r_0 \delta_i} \cdot g_T^{m_i} \\ &= g_T^{r_0 \cdot (\gamma_i \alpha + \delta_i)} \cdot g_T^{m_i} \\ &= g_T^{\alpha_i r_0} \cdot g_T^{m_i} \\ &= e(g_1^{\alpha_i}, g_2)^{r_0} \cdot g_T^{m_i} \\ &= e(H(T_i), g_2)^{r_0} \cdot g_T^{m_i}. \end{aligned}$$

Thus, c_i' has the same distribution as in the SWE security experiment $\mathsf{Exp}_{\mathsf{Sec}}$.

On the other hand, if $(g_1, h_1, v, g_2, h_2, w, z)$ follows the random distribution, then write z as $z = g_T^{\alpha x r + \tau}$ for a uniformly random and independent τ. Since τ is uniformly random, it holds that $\tau \neq 0$, except with negligible probability $1/p$. Thus assume in the following that $\tau \neq 0$. The terms B and $H(T_{\mathsf{ind}})$ are computed as above. The term A is now of the form

$$A = e(v, g_2)^y / z = g_T^{\alpha(rx + r_0)} \cdot g_T^{-\alpha x r - \tau} = g_T^{\alpha r_0 - \tau}.$$

Finally, the terms c_i' for $i \in SC$ are of the form:

$$\begin{aligned} c_i' &= A^{\gamma_i} \cdot B^{\delta_i} \cdot g_T^{m_i} \\ &= g_T^{r_0 \cdot (\gamma_i \alpha + \delta_i) - \tau \gamma_i} \cdot g_T^{m_i} \\ &= g_T^{r_0 \cdot (\gamma_i \alpha + \delta_i)} g_T^{-\tau \gamma} \cdot g_T^{m_i} \end{aligned}$$

Now note that since γ_i and δ_i are uniformly random and independent, $\gamma_i \alpha + \delta_i$ and $\tau \gamma_i$ are also uniformly random and independent as $\tau \neq 0$[4]. Since the term $g_T^{-\tau\gamma}$ is uniformly random and independent of all other terms, it follows that c_i' is uniformly random and thus the output is distributed as in the modified experiment $\mathsf{Exp}'_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$ from the view of $\mathcal{A}$.

If an adversary could distinguish between the two experiments, that therefor implies our distinguisher can decide BDH with the same advantage.

Lastly, note that the output in $\mathsf{Exp}'_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$ is independent of the challenge messages m_i^b. Consequently, in this experiment the advantage of any adversary $\mathcal{A}$ is 0. Security of SWE' follows. □

Theorem 14. *Assume that the hash functions H, H_2, H_{pr} are modelled as random oracles. Then SWE for the signature scheme $\mathsf{Sig}_{\mathsf{agg}}$ is secure under the BDH assumption in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. The security reduction is tight.*

Proof. We have previously shown Lemma 4, which allows us for the base scheme SWE' to replace the ciphertext parts c_i' output in the security experiment, by random elements for all challenge locations, as long as all T_i are distinct.

To leverage this, we first argue that we can blackbox use the outputs of an honest SWE' security challenger (for known plaintext) to faithfully enact an SWE security

[4]This can be seen as the matrix $\begin{pmatrix} \alpha & 1 \\ \tau & 0 \end{pmatrix}$ has full rank given that $\tau \neq 0$.

challenger. Note that SWE' and SWE only differ by SWE additionally providing $a_i = c^{\alpha_i}$ and $t_i = H(T_i)^{\alpha_i}$ and having a modified structure of the ciphertext parts $c'_i = e(H(T_i), g_2)^{r_0\alpha_i} \cdot g_T^{m_i}$. We do this, by asking for the required $H(T_i)$ and encrypting only one challenge message $m_i = m_i^1 = m_i^0$ for each T_i used, which gives us c and c'_i such that we can regain $e(H(T_i), g_2)^{r_0} = c'_i / g_T^{m_i}$. From there, we only need to pick α_i to compute the required a_i, t_i and modified c'_i. This allows us to the apply the lemma and essentially swap out $e(H(T_i), g_2)^{r_0}$ for a random value u_i in that construction and output $c'_i = u_i^{\alpha_i} g_T^{m_i}$.

We notice, that if there are two repeated reference messages $T_i = T_j$, then the resulting ciphertext pieces c'_i in that construction share the same u_i. From there, we use a modified BDH distribution to show that such c'_i, c'_j still look independent and uniformly random based on the fresh choice of α_i.

First, let us define our modified BDH assumption and show, that it follows directly from BDH:

Lemma 5. *Assuming the BDH assumption holds in groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p. Then, the following distributions are computationally close:*

$$(g_1, g_2, g_1^\alpha, g_2^\alpha, g_T^r, g_T^{r\alpha}) \approx_c (g_1, g_2, g_1^\alpha, g_2^\alpha, g_T^r, g_T^u),$$

where $\alpha, r, u \leftarrow\!\!\$\ \mathbb{Z}_p$ and $g_1 \in G_1, g_2 \in G_2, g_T = e(g_1, g_2) \in G_T$ are generators.

The reduction is tight.

Proof. We show that a given distinguisher $\mathcal{D}$ with advantage ε for the above distributions can be used to build a distinguisher against the BDH assumption with the same advantage.

Our BDH distinguisher is given a tuple $(g_1, X_1, Z, g_2, X_2, S, Y)$ which is either distributed as BDH tuple $(g_1, g_1^x, g_1^z, g_2, g_2^x, g_2^s, g_T^{zxs})$ or as random tuple $(g_1, g_1^x, g_1^z, g_2, g_2^x, g_2^s, g_T^y)$, where $x, y, z, s \leftarrow\!\!\$\ \mathbb{Z}_p$.

We input $(g_1, g_2, X_1, X_2, e(Z, S), Y)$ to the distinguisher $\mathcal{D}$ and output what they output. If our input was a BDH tuple, this is distributed as $(g_1, g_2, g_1^x, g_2^x, e(g_1, g_2)^{sz}, e(g_1, g_2)^{zxs})$. This is identically distributed to the left-hand distribution with $\alpha = x, r = sz$ being random values from $\mathbb{Z}_p$.

If our input was a random tuple, this is distributed as $(g_1, g_2, g_1^x, g_2^x, e(g_1, g_2)^{sz}, e(g_1, g_2)^y)$. This is indentically distributed to the right-hand distribution with $\alpha = x, r = sz, u = y$ being random values from $\mathbb{Z}_p$.

Therefore, our distinguishing advantage is identical to that of $\mathcal{D}$ and we conclude, that the distributions are computationally indistinguishable if the BDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

□

Let us assume, that there is an adversary with advantage ε in the security game $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$ for SWE. We define a series of hybrids and claim that they are indistinguishable from the view of $\mathcal{A}$.

- $\mathcal{H}_0$ This is the real game.
- $\mathcal{H}_1$ This experiment behaves exactly as the $\mathcal{H}_\prime$, except, that once we receive the challenge m_i^0, m_i^1 for $i \in SC$, a list of messages $(m_i)_{i \in [\ell] \setminus SC}$ and a list of signing reference messages $(T_i)_{i \in [\ell]}$ from the adversary, the output ct is not created as

  $$\mathsf{ct} = \mathsf{SWE.Enc}(1^\lambda, V, (T_i)_{i \in [\ell]}, (m_i^b)_{i \in [\ell]})$$

  but instead:

- Let $L = \{i \in SC | \forall_{j \in SC : j<i}(T_j \neq T_i)\} \cup \{i \in [\ell] | \forall_{j \in SC \cup [i-1]:}(T_j \neq T_i)\}$ be a subset of indices, such that $(T_i)_{i \in L}$ is a set of all the distinct T_i provided by the adversary (essentially, if some T is ever used at a challenge index $i \in SC$, then the lowest index in SC with that particular T is included in L, otherwise, just the lowest overall index is chosen). We define $\phi(i) = \{j \in L | T_j = T_i\}$ to be the mapping, of the index i of a given T_i into L.
- We run the base algorithm $\mathsf{ct}' = \mathsf{SWE}'.\mathsf{Enc}(1^\lambda, V, (T_i)_{i \in L}, (m_i)_{i \in L})$, where we choose $m_i = m_i^b$ for random $b \leftarrow\!\!\$ \{0,1\}$ for all $i \in SC \cap L$.
- We parse $\mathsf{ct}' = (h, c, c_0, (c_j)_{j \in [n]}, (c'_i)_{i \in L})$.
- We choose random $\alpha_i \leftarrow\!\!\$ \mathbb{Z}_p$ for all $i \in [\ell]$ and determine $H(T_i)$. Then we can set $t_i = H(T_i)^{\alpha_i}, a_i = c^{\alpha_i}$.
- Further, we compute for $i \in L$: $u_i = (c'_i)/g_T^{m_i^b}$
- Then we can set for all $i \in [\ell]$, $c_i^* = u_{\phi(i)}^{\alpha_i} g_T^{m_i^b}$.
- Output $\mathsf{ct} = (h, c, c_0, (c_j)_{j \in [n]}, (c_i^*, a_i, t_i)_{i \in L})$

$\mathcal{H}_2$ This is identical to $\mathcal{H}_1$, except we choose random $u_i \leftarrow\!\!\$ \mathbb{G}_T$ for $i \in SC \cap L$.

$\mathcal{H}_3$ This is identical to $\mathcal{H}_1$, except we choose random $c_i^* \leftarrow\!\!\$ \mathbb{G}_T$ for $i \in SC$.

Claim 25. *$\mathcal{H}_0$ and $\mathcal{H}_1$ are identically distributed.*

Proof. In $\mathcal{H}_0$, the output is $\mathsf{ct} = (h, c = g_2^r, c_0 = h^r g_2^{r_0}, (c_j = \mathsf{vk}_j^r g_2^{s_j})_{j \in [n]}, (c'_i = e(t_i, g_2^{r_0}) \cdot g_t^{m_i^b}, a_i = c^{\alpha_i}, t_i = H(T_i)^{\alpha_i})_{i \in L})$ for random $h \leftarrow \mathbb{G}_2$ and $r, r_0, \alpha_i \leftarrow \mathbb{Z}_p$ and $s_j = f(H_2(\mathsf{vk}_j))$, where f is a random degreee $t-1$ polynomial with $f(0) = r_0$.

In $\mathcal{H}_1$, we receive $\mathsf{ct} = (h, c = g_2^r, c_0 = h^r g_2^{r_0}, (c_j = \mathsf{vk}_j^r g_2^{s_j})_{j \in [n]}, (c'_i = e(H(T_i), g_2^{r_0}) \cdot g_t^{m_i^b})_{i \in L}))$ for random $h \leftarrow \mathbb{G}_2$ and $r, r_0 \leftarrow \mathbb{Z}_p$ and share $s_j = f(H_2(\mathsf{vk}_j))$ as above and output $\mathsf{ct} = (h, c, c_0, (c_j)_{j \in [n]}, (c_i^*, c^{\alpha_i}, t_i = H(T_i)_i^\alpha)_{i \in L})$. By the random choice of α_i, all parts of the distribution agree directly, except we still need to argue that $c_i^* = e(t_i, g_2^{r_0}) \cdot g_t^{m_i^b}$.

By our definition of L and ϕ, we first note, that $c_i^* = u_{\phi(i)}^{\alpha_i} g_T^{m_i^b}$ is well-defined for all $i \in [\ell]$, as we've received u_i for $i \in L$ and ϕ maps directly into L. Now, $c_i^* = ((c'_{\phi(i)})/g_T^{m_{\phi(i)}^b})^{\alpha_i} g_T^{m_i^b}$ By definition of $c'_{\phi(i)}$, this means $c_i^* = (e(H(T_{\phi(i)}), g_0^r)^{\alpha_i} g_T^{m_i^b} = (e(t_i, g_0^r)^{\alpha_i} g_T^{m_i^b}$ and is thus correctly distributed. □

Claim 26. *$\mathcal{H}_1$ and $\mathcal{H}_2$ are indistinguishable given that the BDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.*

Proof. Given that $\mathcal{H}_1$ exactly implements the security game for the base scheme SWE', if we were to directly output ct', thus, we can use Lemma 4 to directly replace all ct'_i for $i \in L \cap SC$ with a random element $u'_i \leftarrow \mathbb{G}_T$. Then, since ct'_i is the only place, where the input m_i is used, we can compute $u_i = u'_i / g_T^{m_i^b}$ as a randomly distributed element for which $\mathsf{ct}'_i = u_i \cdot g_T^{m_i^b}$. □

Claim 27. *$\mathcal{H}_2$ and $\mathcal{H}_3$ are computationally indistinguishable given that the BDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.*

Proof. We reduce distinguishing these hybrids to deciding the modified BDH problem as referenced in Lemma 5 (which tightly reduces to BDH as shown).

Let us receive a tuple $(g_1, g_2, A_1, A_2, Y, Z)$, where $A_1 = g_1^\alpha, A_2 = g_2^\alpha, Y = g_T^y$ for random $y, \alpha \leftarrow\!\!\$ \mathbb{Z}_p$ and Z is either $g_T^{y\alpha}$ or a freshly random g_T^u.

We build a challenger that behaves exactly as in $\mathcal{H}_2$, except in the way, that $H(T_i)$ for $i \in SC$ and ct are chosen.

Let the adversary's challenge be m_i^0, m_i^1 for $i \in SC$, a list of messages $(m_i)_{i\in[\ell]\setminus SC}$ and a list of signing reference messages $(T_i)_{i\in[\ell]}$. We define L and $\phi(i)$ as above. Now, first, we re-randomize our input tuple:

- We choose freshly random $v_i, w_i \leftarrow\!\!\$\ \mathbb{Z}_p$ for $i \in [\ell]$ and $\tau_i, y_i' \leftarrow\!\!\$\ \mathbb{Z}_p$ for $i \in L$. Further set $\tau_i = \tau_{\phi(i)}, y_i' = y'_{\phi(i)}$ for $i \in [\ell] \setminus L$.
- For $i \in L$: $g_{1,i} = g_1^{\tau_i}$, $Y_i = Y^{y_i'}$ and
- For $i \in [\ell]$: $A_{1,i} = (g_1^{v_i} A_1^{w_i})^{\tau_i}$, $A_{2,i} = g_2^{v_i} A_2^{w_i}$, $Z_i = (Y^{v_i} Z^{w_i})^{y_i'}$.

We choose $H(T_i) = g_{1,\phi(i)}$. By definition of ϕ, this ensures that $H(T_i) = H(T_j)$ if $T_i = T_j$, thus making this a well-defined assignment. Further, since we choose $g_{1,\phi(i)} = g_1^{\tau_{\phi(i)}}$, for freshly random $\tau_{\phi(i)}$ for each distinct $T_{\phi(i)}$, $\phi(i) \in L$, this ensures that the output hashes are distributed randomly as in both experiment $\mathcal{H}_2$ and $\mathcal{H}_3$.

The output ct is generated as follows:

- We run the base algorithm $\mathsf{ct}' = \mathsf{SWE}'.\mathsf{Enc}(1^\lambda, V, (T_i)_{i\in L}, (m_i)_{i\in L})$, where we choose $m_i = m_i^b$ for random $b \leftarrow\!\!\$\ \{0,1\}$ for all $i \in SC \cap L$ and gain $\mathsf{ct}' = (h, c = g_2^r, c_0, (c_j)_{j\in[n]}, (c_i')_{i\in L})$.
- For $i \in [\ell] \setminus SC$, we pick random $\alpha_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $a_i = c^{\alpha_i}, t_i = H(T_i)^{\alpha_i}$. For all $i \in [\ell] \cap SC$: we set $t_i = A_{1,i}, a_i = A_{2,i}^r$
- Further we set for $i \in L \setminus SC$ $u_i = (c_i')/g_T^{m_i^b}$ and for $i \in L \cap SC$: $u_i = Y_i$.
- Lastly, we set for $i \in [\ell] \setminus SC$ $c_i^* = u_{\phi(i)}^{\alpha_i} g_T^{m_i^b}$ and for $i \in [\ell] \cap SC$: $c_i^* = Z_i \cdot g_T^{m_i^b}$.
- Output $\mathsf{ct} = (h, c, c_o, (c_j)_{j\in[n]}, (c_i^*, a_i, t_i)_{i\in L})$

Note first: the outputs for indices $i \in [\ell] \setminus SC$ are chosen exactly as in $\mathcal{H}_2$, as these are identically generated also in $\mathcal{H}_3$, there is nothing to show.

Now let us discuss outputs a_i, t_i for $i \in SC$:

$$t_i = A_{1,i} = (g_1^{v_i} A_1^{w_i})^{\tau_i} = (g_1^{\tau_i})^{v_i + w_i\alpha} = H(T_i)^{\tilde{\alpha}_i}$$
$$a_i = A_{2,i}^r = (g_2^{v_i} A_2^{w_i})^r = (g_2^r)^{v_i + w_i\alpha} = c^{\tilde{\alpha}_i}$$

Where $\tilde{\alpha}_i$ is defined as $v_i + w_i\alpha$.

The $u_i = Y_i$ for $i \in L \cap SC$ are distributed as $u_i = Y^{y_i'}$ for a fresh y_i' for each $i \in L \cap SC$ and therefore are randomly distributed as required in $\mathcal{H}_2$. (These inputs aren't actually output and not needed in $\mathcal{H}_3$ at all.)

Now let us regard the outputs c_i^* for $i \in SC$:

- If $Z = g_T^{y\alpha}$, our output is identically distributed to $\mathcal{H}_2$:
  For all $i \in SC$,

$$c_i^*/g_T^{m_i^b} = Z_i = (Y^{v_i} Z^{w_i})^{y_i'} = (g_T^{yv_i} g_T^{y\alpha w_i})^{y_i'}$$
$$= ((g_T^{yy_i'})^{v_i + \alpha w_i}) = (Y^{y_i'})^{\tilde{\alpha}_i} = (Y^{y'_{\phi(i)}})^{\tilde{\alpha}_i} = u_{\phi_i}^{\tilde{\alpha}_i}$$

  This is the same $\tilde{\alpha}_i$ as in a_i, t_i and thus, the output distribution is equivalent to $\mathcal{H}_2$, given that all the $\tilde{\alpha}_i = v_i + w_i\alpha$ are freshly random for each $i \in [\ell]$, given that the v_i, w_i are.
- If we got a random $Z = g_T^u$, our output is identically distributed to $\mathcal{H}_3$:
  We can write $u = y\alpha + \delta$, where δ is a uniquely determined random value derived from Z. Now, for all $i \in SC$,

$$c_i^*/g_T^{m_i^b} = Z_i = (Y^{v_i} Z^{w_i})^{y_i'} = (g_T^{yv_i} g_T^{(y\alpha+\delta)w_i})^{y_i'}$$
$$= ((g_T^{yy_i'})^{v_i + w_i\alpha} \cdot g_T^{\delta w_i y_i'}) = (u_{\phi_i}^{\tilde{\alpha}_i} \cdot g_T^{\delta_i})$$

Where we define $\delta_i = \delta w_i y'_i$. Note that given δ, y'_i for $i \in L$ are freshly random values. Thus, the probability of $\delta y'_i = 0$ is negligible. Then, given we chose for $i \in [\ell]$ fresh v_i, w_i, the derived values $\tilde{\alpha}_i, \delta_i$ for $i \in SC \subseteq [\ell]$ are also two independent randomly distributed values.[5] Thus, as δ_i is independent from all other outputs, we receive, that $c_i^* = u_{\phi_i}^{\tilde{\alpha}_i} \cdot g_T^{\delta_i} g_T^{m_i^b}$ is actually randomly distributed in $\mathbb{G}_T$.

As a result, if an attacker could distinguish these hybrids, we could decide the modified BDH distribution from Lemma 5 with essentially the same advantage.

□

In hybrid $\mathcal{H}_3$, clearly, the advantage of an attacker must be 0, as the outputs do not depend on the challenge messages m_i^b for $i \in SC$ anymore. This concludes the proof. Further we note that all reductions were tight.

□

4.6 A Compatibility Layer for Proof Systems

We will now make our scheme verifiable by constructing a compatibility layer for our SWE scheme and the efficient, ready-to-use Bulletproof [Bün+18] proof system (see § 2.11.5.2).

The high-level idea of this compatibility layer is to first show well-formedness of our ciphertexts and then attach a bulletproof-compatible Pedersen commitment to each ciphertext. We provide an efficient NIZK proof guaranteeing that ciphertext and commitment encrypt the same value. We can then use Bulletproofs to establish any additional properties about the encrypted message.

We focus on the enhanced SWE, as it is slightly more complex. All the proofs work analogously for the base SWE' and we give a few comments, where appropriate, on how they need to be modified to apply to the base scheme.

We will once again implicitly assume that all algorithms in this section have oracle access to full-domain hash functions $H : \{0,1\}^* \to \mathbb{G}_1$ and $H_2, H_{pr} : \{0,1\}^* \to \mathbb{Z}_p$ as used in Construction 4 and Construction 5.

4.6.1 Well-Formedness Proofs

In this section we will provide a proof system to efficiently prove that a given SWE ciphertext is decryptable. More precisely, this proof system will ensure that the SWE scheme is committing in the sense that regardless of which committee-members contribute to the multi-signature, the decrypted message is always the same. This proof system will not yet ensure that the message m_i are in the correct range for discrete log extraction, though. This will be ensured using the proof systems in § 4.6.2 and § 4.6.3.

We will provide a proof system to certify that ciphertexts generated by $\mathsf{SWE.Enc}$ are well-formed. That is, we define a NIZK proof $(\mathsf{P}_1, \mathsf{V}_1)$ for the language $\mathcal{L}_1$ defined by the relation

$$(x = (\mathsf{ct}, (\mathsf{vk}_j)_{j\in[n]}, (T_i)_{i\in[\ell]}), w = ((m_i)_{i\in[\ell]}, r_1)) \in \mathcal{R}_1 \Leftrightarrow$$
$$\mathsf{ct} = \mathsf{SWE.Enc}(1^\lambda, (\mathsf{vk}_j)_{j\in[n]}, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]}; r_1).$$

[5] This can again be seen as the matrix $\begin{pmatrix} \alpha & 1 \\ y'_i\delta & 0 \end{pmatrix}$ has full rank given that $y'_i\delta \neq 0$.

The ciphertexts produced by $\mathsf{SWE.Enc}$ are of the form $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c'_i, a_i, t_i)_{i\in[\ell]})$ as follows:

$$\begin{aligned} c &= g_2^r \\ c_0 &= h^r \cdot g_2^{r_0} \\ c_j &= \mathsf{vk}_j^r \cdot g_2^{s_j} \text{ for } j = 1, \ldots, n \\ a_i &= c^{\alpha_i} \text{ for } i = 1, \ldots, \ell \\ t_i &= H(T_i)^{\alpha_i} \text{ for } i = 1, \ldots, \ell \\ c'_i &= e(t_i, g_2)^{r_0} \cdot g_T^{m_i} \text{ for } i = 1, \ldots, \ell \end{aligned}$$

where $(s_1, \ldots, s_n)$ is the vector of shares of r_0 under Shamir's secret sharing scheme. That is, a random polynomial $f(x)$ with $f(0) = r_0$ is sampled, and the shares are chosen as $s_i = f(\xi_i)$ for distinct interpolation points ξ_i, so r_0 can be retrieved from polynomial interpolation. We introduced Reed-Solomon codes and their relation with Shamir's sharings in § 2.9.2. It holds that $\boldsymbol{s} = (r_0, s_1, \ldots, s_n)$ is correctly formed as a set of shares s_i over the message r_0 if and only if $\boldsymbol{s}$ is a codeword of the Reed-Solomon code $\mathsf{RS}_{n+1,t}[\boldsymbol{\xi}]$ with support points $\boldsymbol{\xi} = (0, \xi_1, \ldots, \xi_n)$, which is equivalent to the equation $\mathbf{H} \cdot \boldsymbol{s} = 0$ holding, where $\mathbf{H} \in \mathbb{Z}_p^{(n+1-t)\times(n+1)}$ is the parity-check matrix of $\mathsf{RS}_{n+1,t}[\boldsymbol{\xi}]$.

To prove well-formedness of such a ciphertext, we proceed as follows. Let H' and H'' be hash-functions (modelled as a random oracle).

$\mathsf{P}_1(\mathsf{ct}, (\mathsf{vk}_j)_j, r)$: The prover proceeds as follows, requiring only $r \in \mathbb{Z}_p$ as witness.

- Compute $\boldsymbol{v} = H'(\mathsf{ct}) \in \mathbb{Z}_p^{n+1-t}$ and set $\boldsymbol{w}^\top = \boldsymbol{v}^\top \cdot \mathbf{H}$.
- Parse $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$.
- Set $c^* = c_0^{w_0} \cdot \prod_{j=1}^n c_j^{w_j}$.
- Set $g^* = h^{w_0} \cdot \prod_{j=1}^n \mathsf{vk}_j^{w_j}$.
- Choose $y \leftarrow \mathbb{Z}_p$ uniformly at random and set $f = g_2^y$ and $f^* = (g^*)^y$.
- Compute $\alpha = H''(g_2, c, g^*, c^*, f, f^*)$.
- Compute $z = y + \alpha r$.
- Output $\pi = (f, f^*, z)$.

$\mathsf{V}_1(\mathsf{ct}, (\mathsf{vk}_j)_j, \pi)$: To verify a proof $\pi = (f, f^*, z)$, proceed as follows.

- Check if $e(t_i, c) = e(H(T_i), a_i)$ for all $i \in [\ell]$. Otherwise, output 0.
- Compute $\boldsymbol{v} = H'(\mathsf{ct})$ and set $\boldsymbol{w} = \boldsymbol{v}^\top \cdot \mathbf{H}$. Parse $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$.
- Set $c^* = c_0^{w_0} \cdot \prod_{j=1}^n c_j^{w_j}$.
- Set $g^* = h^{w_0} \cdot \prod_{j=1}^n \mathsf{vk}_j^{w_j}$.
- Compute $\alpha = H''(g_2, c, g^*, c^*, f, f^*)$.
- Check if $(c^*)^\alpha \cdot (f^*) = (g^*)^z$ and $c^\alpha \cdot f = g_2^z$, if so output 1, otherwise 0.

For the base scheme SWE', we have essentially the same scheme, but omit a_i, t_i, as all α_i are set to 1. The proof system works exactly the same, except that we omit the first check of $e(t_i, c) = e(H(T_i), a_i)$ in verification, which is made to ensure that $(c, a_i), (H(T_i), t_i)$ share the same dlog relationship to allow decryption.

Theorem 15. *$(\mathsf{P}_1, \mathsf{V}_1)$ is a NIZK proof system for relation $\mathcal{R}_1$ assuming H', H'' are modelled as random oracles. Concretely, if $x = (\mathsf{ct}, (\mathsf{vk}_j)_{j\in[n]}) \notin \mathcal{L}_1$ and P_1 makes at most q queries to either H' or H'', then V_1 rejects x, except with negligible probability q/p.*

Proof. We argue Completeness, Soundness and Zero-Knowledge of this scheme.

Completeness. Perfect Completeness of this proof system follows routinely.

For all $i \in [\ell]$, $e(t_i, c) = e(H(T_i)^{\alpha_i}, c) = e(H(T_i), c^{\alpha_i}) = e(H(T_i), a_i)$ holds by construction.

We observe the verifier constructs the same values as the prover for $\boldsymbol{v}, \boldsymbol{w}, c^*, g^*, \alpha$. Now, assuming the input was such that $\mathsf{ct} = \mathsf{SWE.Enc}(1^\lambda, (\mathsf{vk}_j)_j, \cdot, \cdot; r_1)$, where r_1 contains r as the random exponent used in ciphertext part c, then it holds

$$\begin{aligned}(c^*)^\alpha \cdot (f^*) &= (c_0^{w_0} \cdot \prod_{j=1}^{n} c_j^{w_j})^\alpha \cdot (g^*)^y \\ &= ((h^r \cdot g_2^{r_0})^{w_0} \cdot \prod_{j=1}^{n} (\mathsf{vk}_j^r \cdot g_2^{s_j})^{w_j})^\alpha \cdot (g^*)^y \\ &= g_2^{r_0 w_0 \alpha} \prod_{j=1}^{n} g_2^{s_j w_j \alpha} \cdot (g^*)^{y + r\alpha} = (g^*)^z\end{aligned}$$

since $\boldsymbol{w}^\top (r_0, s_1, \ldots, s_n) = \boldsymbol{v}^\top \cdot \mathbf{H}(r_0, s_1, \ldots, s_n) = 0$ as discussed above. The other equation checked in V_1 can be shown to hold analogously.

Soundness. To show computational soundness, first note the following: A ciphertext ct is well-formed if and only if

1. $\boldsymbol{s} = (r_0, s_1, \ldots, s_n) \in \mathsf{RS}[\mathbb{Z}_q, n+1, t]$ and
2. $(c, a_i), (H(T_i), t_i)$ share the same dlog relationship.

(2) is directly implied by the check $e(t_i, c) = e(H(T_i), a_i)$. We focus on the first condition: It is exactly fulfilled if $\mathbf{H}\boldsymbol{s} = 0$. Thus assume that ct is not a valid ciphertext, i.e. it holds that $\mathbf{H}\boldsymbol{s} \neq 0$. Say that a challenge $\boldsymbol{v}$ is *bad*, if it holds that $\boldsymbol{v}^\top \mathbf{H}\boldsymbol{s} = 0$. Since $\boldsymbol{v}$ is chosen uniformly random and $\mathbf{H}\boldsymbol{s} \neq 0$ it holds that

$$\Pr[\boldsymbol{v} \text{ bad}] = \Pr[\boldsymbol{w}^\top \boldsymbol{s} = 0] = \Pr[\boldsymbol{v}^\top (\mathbf{H}\boldsymbol{s}) = 0] = \frac{1}{p},$$

which is negligible over the random choice of $\boldsymbol{v}$. Note that it holds that

$$c^* = c_0^{w_0} \prod_{j=1}^{n} c_j^{w_j} = h^{r w_0} \cdot g_2^{r_0 w_0} \prod_{j=1}^{n} (\mathsf{vk}_j^r \cdot g_2^{s_j})^{w_j} = (g^*)^r \cdot g_2^{\boldsymbol{w}^\top \boldsymbol{s}}.$$

Now fix a $\boldsymbol{v}$, which is not bad, i.e. it holds that $\boldsymbol{w}^\top \boldsymbol{s} \neq 0$. Then it holds that $(g_2, c = g_2^r)$ and $(g^*, c^* = (g^*)^r \cdot g_T^{\boldsymbol{w}^\top \boldsymbol{s}})$ do not satisfy the same discrete logarithm relation, in other words the vectors $(1, \log_{g_2}(c))$ and $(1, \log_{g^*}(c^*))$ are linearly independent. But this means that also $(1, 1)$ and $(\log_{g_2}(c), \log_{g^*}(c^*))$ are linearly independent.

Now fix any f and f^* which may depend on $\boldsymbol{v}$. Say that an α is *bad*, if there exists a $z \in \mathbb{Z}_p$ such that

$$\begin{aligned}c^\alpha \cdot f &= g_2^z \\ (c^*)^\alpha \cdot f^* &= (g^*)^z.\end{aligned}$$

Now observe that α is bad if and only if

$$\alpha(\log_{g_2}(c), \log_{g^*}(c^*)) + (\log_{g_2}(f), \log_{g*}(f^*)) \in \mathsf{Span}((1,1)),$$

which is equivalent to

$$\alpha(\log_{g_2}(c), \log_{g^*}(c^*)) \in -(\log_{g_2}(f), \log_{g^*}(f^*)) + \mathsf{Span}((1,1)).$$

As $(1,1)$ and $(\log_{g_2}(c), \log_{g^*}(c^*))$ are linearly independent, $\alpha(\log_{g_2}(c), \log_{g^*}(c^*))$ only lands in the affine subspace $-(\log_{g_2}(f), \log_{g^*}(f^*)) + \mathsf{Span}((1,1))$ with negligible probability $1/p$ over the choice of α. It follows that $\Pr[\alpha \text{ bad}] \leq 1/p$.

Now note that V_1 rejects if neither $\boldsymbol{v}$ nor α is bad. Now assume that the adversary makes q_1 queries to H' and q_2 queries to H''. By a union bound we conclude the the probability that one of the queries to H' results in a bad $\boldsymbol{v}$ is at most q_1/p. Furthermore, also by a union bound the probability that one of the queries to H'' results in a bad α is at most q_2/p. We conclude with a union bound that there are no bad queries to H' or H'', except with probability $(q_1+q_2)/p$, which is negligible.

Zero-Knowledge. To show that this proof-system is zero-knowledge, we can routinely make use of the fact that the underlying Schnorr-like proof-system for equality of discrete logarithms (see e.g. [CKY09]) is zero-knowledge.

Specifically, for every statement to prove, our simulator chooses z and α uniformly at random and sets $f = g_2^z \cdot c^{-\alpha}$ and $f^* = (g^*)^z \cdot (c^*)^{-\alpha}$ and programs the random oracle H'' to output α on the query $(g_2, c, g^*, c^*, f, f^*)$.

It follows routinely that given a YES-instance $(\mathsf{ct}, (\mathsf{vk}_j)_j)$ the simulation is perfect unless the verifier makes a hash query $(g_2, c, g^*, c^*, f, f^*)$ to H'' that the simulator would make, thus prohibiting the simulator to program H'' accordingly. However, since f is distributed uniformly at random, this happens only with negligible probability $1/p$. Since the randomness is freshly drawn for each statement, simulation of a polynomial number of statements still only fails with a negligible chance. Thus we have established the zero-knowledge property.

□

4.6.2 Proofs of Plaintext Equality

First observe the following: A ciphertext $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c'_i, a_i, t_i)_{i\in[\ell]})$ is a perfectly binding, computationally hiding and homomorphic commitment for messages $(m_i)_i \in \mathbb{Z}_p^\ell$, even if we drop the c_j, a_i. In fact, the only purpose of the c_j and a_i is to facilitate decryption.

Regarding SWE Ciphertexts as Homomorphic Commitments. To be precise, we will call this commitment com and following Def. 20, we can fix randomly h and the $h_{T,i} = e(t_i, g_2)$ as the CRS crs output by $\mathsf{com.Setup}$. Then the commitment scheme will output $\mathsf{com.Commit}(\mathsf{crs}, \boldsymbol{m}; r, r_0) = (c = g_2^r, c_0 = h^r \cdot g_2^{r_0}, (c'_i = h_{T,i}^{r_0} \cdot g_T^{m_i})_i)$ with opening (r, r_0). Given this opening and a message vector $\boldsymbol{m}$, $\mathsf{com.Vrfy}$ can check whether a commitment adheres to that format and in that case considers it valid.

Correctness of this scheme is immediately clear. The hiding property can be reduced to BDH in a similar manner as in our IND-CPA security proofs of SWE', where we show that $h_{T,i}^{r_0}$ can be replaced by fresh random values and therefore hides $g_T^{m_i}$. This is done by chosing the $g_2^{r_0}, h_{T,i}, h_{T,i}^{r_0}$ from rerandomized BDH challenge values. The intuition to show the binding property is as follows: If there are $r, r_0, \boldsymbol{m}, r', r'_0, \boldsymbol{m}'$ such that $\mathsf{com}(\boldsymbol{m}; r, r_0) = \mathsf{com}(\boldsymbol{m}'; r', r'_0)$, then as g_2 is a generator and $c = g_2^r$, $r = r'$. Then, h^r becomes fixed and from $c_0 = h^r \cdot g_2^{r_0}$, we conclude $r_0 = r'_0$. The same holds for all m_i individually as g_T is a generator.

This commitment scheme is homomorphic, i.e. it holds

$$\mathsf{com.Commit}(\mathsf{crs}, \boldsymbol{m}; r, r_0)^\alpha \cdot \mathsf{com.Commit}(\mathsf{crs}, \boldsymbol{m}'; r', r'_0)^\beta = \mathsf{com.Commit}(\mathsf{crs}, \alpha\boldsymbol{m} + \beta\boldsymbol{m}'; \alpha r + \beta r', \alpha r_0 + \beta r'_0),$$

where the multiplication and exponentiation of commitments is component-wise. Homomorphic operations for this commitment scheme are relatively expensive, as the c'_i components reside in the target group $\mathbb{G}_T$.

Adding Pedersen Commitments. To enable efficient proofs of statements over the commited values $\boldsymbol{m}$, we will leverage a second commitment scheme COM and provide a highly efficient cross-scheme plaintext equality proof for two commitments $\mathsf{com}(\boldsymbol{m})$ and $\mathsf{COM}(\boldsymbol{m})$. Furthermore, we will choose the commitment scheme COM to be proof-system friendly. Thus, the natural choice for COM is a Pedersen commitment in a cryptographic group $\mathbb{G}$ of order p, as introduced in § 2.10.3.2. For simplicity, we may choose e.g. $\mathbb{G} = \mathbb{G}_1$, but $\mathbb{G}$ could in fact be any group of order p.

Let $g, h \leftarrow \mathbb{G}$ be public but randomly chosen group elements from the group $\mathbb{G}$ and $\mathsf{CRS} = (g, h)$. Then a Pedersen (scalar) commitment for a vector of messages $\boldsymbol{m}$ is computed by $\mathsf{COM.Commit}(\mathsf{CRS}, \boldsymbol{m}; \boldsymbol{\rho}) = (g^{\rho_i} \cdot h^{m_i})_{i\in[\ell]}$ for uniformly random values $\rho_i \leftarrow \mathbb{Z}_p$. This commitment is computationally binding and perfectly hiding under the discrete logarithm assumption in $\mathbb{G}$. It is homomorphic in the same sense as com.

We will now provide a non-interactive zero-knowledge proof of knowledge $(\mathsf{P}_2, \mathsf{V}_2)$ for the following relation $\mathcal{R}_2$ that is dependent on given $\mathsf{crs} = (h \in \mathbb{G}_2, h_{T,1}, \ldots, h_{T,\ell} \in \mathbb{G}_T)$ and $\mathsf{CRS} = (g, h \in \mathbb{G})$:

$$(x = (\mathsf{c}, \mathsf{C}), w = (\boldsymbol{m}, r, r_0, \boldsymbol{\rho})) \in \mathcal{R}_2^{\mathsf{crs},\mathsf{CRS}} \Leftrightarrow$$
$$\mathsf{c} = \mathsf{com.Commit}(\mathsf{crs}, \boldsymbol{m}; r, r_0) \text{ and } \mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, \boldsymbol{m}; \boldsymbol{\rho}).$$

Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a hash-function, which will be modelled as a random oracle.

$\mathsf{P}_2((\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}), (\boldsymbol{m}, r, r_0, \boldsymbol{\rho}))$:

- Choose $\boldsymbol{u}, \boldsymbol{\rho'} \leftarrow \mathbb{Z}_p^\ell$ uniformly at random.
- Choose $r'_0, r' \leftarrow \mathbb{Z}_p$ uniformly at random.
- Compute $\mathsf{c}' = \mathsf{com.Commit}(\mathsf{crs}, \boldsymbol{u}; r', r'_0)$.
- Compute $\mathsf{C}' = \mathsf{COM.Commit}(\mathsf{CRS}, \boldsymbol{u}; \boldsymbol{\rho'})$.
- Compute $\alpha = H(\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}, \mathsf{c}', \mathsf{C}')$.
- Compute $\tilde{\boldsymbol{m}} = \boldsymbol{u} + \alpha\boldsymbol{m}$, $\tilde{r} = r' + \alpha r$, $\tilde{r}_0 = r'_0 + \alpha . r_0$ and $\tilde{\boldsymbol{\rho}} = \boldsymbol{\rho'} + \alpha\boldsymbol{\rho}$.
- Output $\pi = (\mathsf{c}', \mathsf{C}', \tilde{\boldsymbol{m}}, \tilde{r}, \tilde{r}_0, \tilde{\boldsymbol{\rho}})$.

$\mathsf{V}_2((\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}), \pi = (\mathsf{c}', \mathsf{C}', \tilde{\boldsymbol{m}}, \tilde{r}, \tilde{r}_0, \tilde{\boldsymbol{\rho}}))$:

- Compute $\alpha = H(\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}, \mathsf{c}', \mathsf{C}')$.
- Check if $\mathsf{c}' \cdot \mathsf{c}^\alpha \stackrel{?}{=} \mathsf{com.Commit}(\mathsf{crs}, \tilde{\boldsymbol{m}}; \tilde{r}, \tilde{r}_0)$
- Check if $\mathsf{C}' \cdot \mathsf{C}^\alpha \stackrel{?}{=} \mathsf{COM.Commit}(\mathsf{CRS}, \tilde{\boldsymbol{m}}; \tilde{\boldsymbol{\rho}})$
- If both checks are true, output 1, otherwise 0.

We note that we can mount the same argument and run the exact same proof system for the base scheme $\mathsf{SWE'}$, by setting $h_{T,i} = e(H(T_i), g_2))$ instead, as this results in the exact same commitment scheme $\mathsf{com.Commit}(\mathsf{crs} = (h, (h_{T,i})_i), \boldsymbol{m}; r, r_0) = (c = g_2^r, c_0 = h^r \cdot g_2^{r_0}, (c'_i = h_{T,i}^{r_0} \cdot g_T^{m_i})_i)$ being a subset of $\mathsf{SWE'}$ ciphertexts.

Theorem 16. $(\mathsf{P}_2, \mathsf{V}_2)$ *is a NIZK proof of knowledge for relation* $\mathcal{R}_2$*, assuming* H *is modelled as random oracle.*

Proof. We argue the properties of completeness, zero-knowledge and weak simulation extractability. Note that our definition of a NIZKPoK, Def. 36, is defined in the random oracle model, whereas this protocol takes a CRS as input. This is not a

problem, as a CRS can conceptually be drawn freshly randomly from a specific seed queried against the random oracle, we will consider the CRS to be given in such a publicly fixing way in the following.

Completeness. Perfect completeness of this proof system follows routinely. On an honestly generated input $((\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}), \pi = \mathsf{P}_2((\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}), (\boldsymbol{m}, r, r_0, \boldsymbol{\rho})))$ to V_2, we can see that $\mathsf{c}' \cdot \mathsf{c}^{\alpha} = \mathsf{com.Commit}(\mathsf{crs}, \boldsymbol{u}; r', r'_0) \cdot \mathsf{com.Commit}(\mathsf{crs}, \boldsymbol{m}; r, r_0)^{\alpha} = \mathsf{com.Commit}(\mathsf{crs}, \tilde{\boldsymbol{m}}; \tilde{r}, \tilde{r}_0)$ by our definitions of $\tilde{\boldsymbol{m}}, \tilde{r}, \tilde{r_0}$ and the underlying scheme being homomorphic. The same holds for $\mathsf{C}' \cdot \mathsf{C}^{\alpha} \stackrel{?}{=} \mathsf{COM.Commit}(\mathsf{CRS}, \tilde{\boldsymbol{m}}; \tilde{\boldsymbol{\rho}})$, so on an honest input, P_2 always outputs 1.

Zero-Knowledge. To argue that $(\mathsf{P}_2, \mathsf{V}_2)$ is zero-knowledge, we construct a simulator $\mathcal{S}$ which, given $\mathsf{crs}, \mathsf{CRS}$ and a statement (c, C), chooses uniformly random $\tilde{\boldsymbol{m}}, \boldsymbol{\rho} \leftarrow \mathbb{Z}_p^{\ell}$, $\tilde{r}, \tilde{r}_0 \leftarrow \mathbb{Z}_p$ as well as a uniformly random $\alpha \leftarrow \mathbb{Z}_p$ and sets $\mathsf{c}' = \mathsf{com.Commit}(\mathsf{crs}, \tilde{\boldsymbol{m}}; \tilde{r}, \tilde{r}_0) \cdot \mathsf{c}^{-\alpha}$ and $\mathsf{C}' = \mathsf{COM.Commit}(\mathsf{CRS}, \tilde{\boldsymbol{m}}; \tilde{\boldsymbol{\rho}}) \cdot \mathsf{C}^{-\alpha}$. Further, the simulator $\mathcal{S}$ programs the random oracle H to output α on input $(\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}, \mathsf{c}', \mathsf{C}')$. The simulated proof π is given by $\pi = (\mathsf{c}', \mathsf{C}', \tilde{\boldsymbol{m}}, \tilde{r}, \tilde{r}_0, \tilde{\boldsymbol{\rho}})$.

It follows routinely that the simulation is perfect, unless $\mathcal{A}$ queries H on $(\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}, \mathsf{c}', \mathsf{C}')$ before obtaining the proof π. But this only happens with negligible probability as the commitments c' and C' are freshly chosen by $\mathcal{S}$.

Proof of Knowledge. We will show weak simulation extractability of $(\mathsf{P}_2, \mathsf{V}_2)$. That is, for any adversary that outputs an accepting proof for a true statement, there exists an efficient extractor that can recover a corresponding witness. Our strategy uses the Expected-Time Forking Lemma [BN06] (see Lemma 1), which allows us to extract the witness by rewinding the adversary in the random oracle model.

Let $\mathsf{crs}, \mathsf{CRS}$ be given common-reference strings for $\mathsf{com}, \mathsf{COM}$. Let P_2^* be a malicious prover with non-negligible advantage ν. That is, we give it $(\mathsf{CRS}, \mathsf{crs})$ and allow it to receive simulated proofs π_i for statement-witness pairs (x_i, w_i) of its choice as well as oracle access to a hash H, which is programmed by the zero-knowledge simulator. In the end it must output a statement $x^* = (\mathsf{c}, \mathsf{C})$ and verifying proof $\pi^* = (\mathsf{c}', \mathsf{C}', \cdot, \cdot, \cdot, \cdot)$, such that (x^*, π^*) was not received from the experiment. Critically, one of the hash queries made by P_2^* must have been $\alpha = H(\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}, \mathsf{c}', \mathsf{C}')$, in order to give a correct proof.

We wish to rewind execution of P_2^* such that we get a fork consisting of two verifying proofs $(\pi = (\mathsf{c}', \mathsf{C}', \tilde{\boldsymbol{m}}, \tilde{r}, \tilde{r}_0, \tilde{\boldsymbol{\rho}}), \pi' = (\mathsf{c}', \mathsf{C}', \hat{\boldsymbol{m}}, \hat{r}, \hat{r}_0, \hat{\boldsymbol{\rho}}))$ for the same statement (c, C), using the same c', C', but different hash outputs α, α' provided on query $H(\mathsf{crs}, \mathsf{CRS}, \mathsf{c}, \mathsf{C}, \mathsf{c}', \mathsf{C}')$.

Lemma 1 guarantees that, since P_2^* runs in polynomial time, makes only polynomially many hash queries and can produce a valid proof with non-negligible probability ν, then there exists a rewinding strategy in form of a forking algorithm $\mathsf{F}_{\mathsf{P}_2^*}$, which runs in expected polynomial time and which, with the same probability ν, produces such a fork (π, π').

Based on this, we give a knowledge extractor $\mathcal{E}$ as follows:

- Run $\mathsf{F}_{\mathsf{P}_2^*}$, and if it outputs 0, also output 0.
- Otherwise, if $\mathsf{F}_{\mathsf{P}_2^*}$ outputs a fork (π, α) and (π', α') proceed as follows.
- Parse $\pi = (\mathsf{c}', \mathsf{C}', \tilde{\boldsymbol{m}}, \tilde{r}, \tilde{r}_0, \tilde{\boldsymbol{\rho}})$ and $\pi' = (\mathsf{c}', \mathsf{C}', \hat{\boldsymbol{m}}, \hat{r}, \hat{r}_0, \hat{\boldsymbol{\rho}})$.
- Compute $\bar{\boldsymbol{m}} = (\tilde{\boldsymbol{m}} - \hat{\boldsymbol{m}})/(\alpha - \alpha')$, $\bar{r} = (\tilde{r} - \hat{r})/(\alpha - \alpha')$, $\bar{r}_0 = (\tilde{r}_0 - \hat{r}_0)/(\alpha - \alpha')$ and $\bar{\boldsymbol{\rho}} = (\tilde{\boldsymbol{\rho}} - \hat{\boldsymbol{\rho}})/(\alpha - \alpha')$ and output $w = (\bar{\boldsymbol{m}}, \bar{r}, \bar{r}_0, \bar{\boldsymbol{\rho}})$.

Let us now argue our success probability of outputting a correct witness w. In case that π, π' verify for statement $x^* = (\mathsf{c}', \mathsf{C}')$ it holds by our definition of V_2 that:

$$\mathsf{C}' \cdot \mathsf{C}^{\alpha} = \mathsf{COM.Commit}(\mathsf{CRS}, \tilde{\boldsymbol{m}}; \tilde{\boldsymbol{\rho}}) \text{ and } \mathsf{C}' \cdot \mathsf{C}^{\alpha'} = \mathsf{COM.Commit}(\mathsf{CRS}, \hat{\boldsymbol{m}}; \hat{\boldsymbol{\rho}}),$$

from which we can conclude by homomorphism of COM that

$$\mathsf{C}^{\alpha-\alpha'} = \mathsf{COM.Commit}(\mathsf{CRS}, \tilde{\boldsymbol{m}} - \hat{\boldsymbol{m}}; \tilde{\boldsymbol{\rho}} - \hat{\boldsymbol{\rho}}),$$

and therefore

$$\begin{aligned}\mathsf{C} &= \mathsf{COM.Commit}(\mathsf{CRS}, (\tilde{\boldsymbol{m}} - \hat{\boldsymbol{m}})/(\alpha - \alpha'); (\tilde{\boldsymbol{\rho}} - \hat{\boldsymbol{\rho}})/(\alpha - \alpha')) \\ &= \mathsf{COM.Commit}(\mathsf{CRS}, \bar{\boldsymbol{m}}; \bar{\boldsymbol{\rho}}).\end{aligned}$$

An analogous argument can be mounted to show that $\mathsf{c} = \mathsf{com.Commit}(\mathsf{crs}, \bar{\boldsymbol{m}}; \bar{r}, \bar{r_0})$. This means that $(x^*, w) \in \mathcal{R}_2^{\mathsf{crs},\mathsf{CRS}}$ and the witness output by $\mathcal{E}$ is correct, whenever the internally run $\mathsf{F}_{\mathsf{P}_2^*}$ produces a fork.

So, $\mathcal{E}$ is an algorithm with expected polynomial runtime and extraction probability ν, thus our knowledge extractor $\mathcal{E}$ is efficient and correct.

□

4.6.3 Putting Everything Together: Verifiable SWE

We will now briefly discuss how we can extend the SWE scheme constructed in § 4.5.1 with the proof systems of this section and an efficient proof system (e.g. Bulletproofs [Bün+18]) to obtain an efficient *verifiable* SWE scheme as defined in Def. 70, i.e., an SWE for which we can efficiently prove statements about the encrypted messages.

Let any relation efficiently checkable $\mathcal{R}$ for messages m and witnesses w be given.

In the following, we extend SWE with two algorithms $\mathsf{SWE.Prove}$, $\mathsf{SWE.Vrfy}$ which form a NIZK proof system in the ROM for the language given by the induced relation $\mathcal{R}'$:

$$\begin{aligned}(V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n), (T_i)_{i\in[\ell]}, \mathsf{ct}), ((m_i)_{i\in[\ell]}, w, r')) \in \mathcal{R}' \Leftrightarrow \\ \mathsf{ct} = \mathsf{SWE.Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]}); r') \text{ and } \forall i \in [\ell] : m_i \in \{0, \ldots, 2^k - 1\} \\ \text{and } (m = \sum_{i\in[\ell]} 2^{(i-1)k} m_i, w) \in \mathcal{R}\end{aligned}$$

To this end, let

- $(\mathsf{P}_1, \mathsf{V}_1)$ be the NIZK proof system for well-formedness constructed § 4.6.1.
- $(\mathsf{P}_2, \mathsf{V}_2)$ be the NIZKPoK proof system for plaintext equality constructed in § 4.6.2.
- $(\mathsf{P}_3, \mathsf{V}_3)$ be a NIZKPoK proof system which asserts for a given Pedersen commitment $\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, (m_i)_{i\in[\ell]}; \boldsymbol{\rho})$ that $\forall i \in [\ell] : m_i \in \{0, \ldots, 2^k - 1\}$ and $(m = \sum_{i\in[\ell]} 2^{(i-1)k} m_i, w) \in \mathcal{R}$.

Bulletproofs. We choose Bulletproofs to instantiate $(\mathsf{P}_3, \mathsf{V}_3)$, as they are compatible with the Pedersen commitments COM, as outlined in § 2.11.5.2. The format of $(\mathsf{P}_3, \mathsf{V}_3)$ is as follows: It takes as input a CRS, a commitment to a vector $\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, \boldsymbol{m}; \boldsymbol{\rho})$ and it takes as witness $\boldsymbol{m}, \boldsymbol{\rho}$ and an auxiliary witness w, which establishes $(\sum_{i\in[\ell]} 2^{(i-1)k} m_i, w) \in \mathcal{R}$. In the following, for succinctness, we assume that the commitment scheme COM takes the same common reference string

CRS as provided for $(\mathsf{P}_3, \mathsf{V}_3)$ and we once again implicitly require that the CRS is chosen from the random oracle.

We actually combine two different flavours of bulletproofs to achieve this. We require $(\mathsf{P}_3, \mathsf{V}_3)$ to show two different properties: Firstly, recall that we must enforce that each component m_i is in the appropriate range, i.e. $m_i \in \{0, \ldots, 2^k - 1\}$ for a small integer k, to ensure efficient decryption of our SWE ciphertexts, even for maliciously generated ciphertexts, using the baby-step giant-step algorithm. The range-proof variant of bulletproofs ensures that this can be shown for all messages m_i with a single proof of size $\mathcal{O}(\log_2(k \cdot \ell))$, compare Lemma 3. Secondly, for additional, more complex relations $\mathcal{R}$, we rely on the more general Bulletproof construction for circuit-based relations. If $\mathcal{R}$ can be represented by a circuit of multiplicative complexity N, then the proof of knowledge establishing that C indeed commits to messages $\boldsymbol{m}$ with $(\sum_{i\in[\ell]} 2^{(i-1)k} m_i, w) \in \mathcal{R}$ for a witness w is of size $\mathcal{O}(\log_2(N))$, compare Lemma 2.

Our Construction. We require the random coins r' used in SWE encryption to be input to the proof, and then extract the subset of random coins (r, r_0) as required by $(\mathsf{P}_1, \mathsf{V}_1)$ and $(\mathsf{P}_2, \mathsf{V}_2)$. Note that executing encryption and the proof in one instance is more efficient and preferred in practice.

Construction 7. *We describe* $\mathsf{Prove}, \mathsf{Vrfy}$ *that extend* SWE *with* $\mathsf{Enc}, \mathsf{Dec}$ *as described in § 4.5.1 to be a verifiable SWE scheme.*

$\mathsf{SWE.Prove}(\mathsf{CRS}, (\mathsf{vk}_j)_{j\in[n]}, (T_i)_{i\in[\ell]}, \mathsf{ct}, (m_i)_{i\in[\ell]}, w, r') :$

- *Pick necessary random coins* r, r_0 *from* r'.
- *Parse* $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c'_i, a_i, t_i)_{i\in[\ell]})$.
- *Set* $h_{T,i} = e(t_i, g_2)$ *for* $i \in [\ell]$.
- *Choose* $\boldsymbol{\rho} \leftarrow \mathbb{Z}_p^\ell$ *uniformly at random.*
- *Compute* $\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, (m_i)_{i\in[\ell]}; \boldsymbol{\rho})$.
- *Compute* $\pi_1 = \mathsf{P}_1(\mathsf{ct}, (\mathsf{vk}_j)_j, r)$.
- *Set* $\mathsf{crs} = (h, (h_{T,i})_{i\in[\ell]})$.
- *Compute* $\pi_2 = \mathsf{P}_2((\mathsf{crs}, \mathsf{CRS}, (c, c_0, (c'_i)_{i\in[\ell]}), \mathsf{C}), ((m_i)_{i\in[\ell]}, r, r_0, \boldsymbol{\rho}))$.
- *Compute* $\pi_3 = \mathsf{P}_3(\mathsf{CRS}, \mathsf{C}, (\boldsymbol{m}, \boldsymbol{\rho}, w))$.
- *Output* $\pi = (\mathsf{C}, \pi_1, \pi_2, \pi_3)$.

$\mathsf{SWE.Vrfy}(\mathsf{CRS}, (\mathsf{vk}_j)_{j\in[n]}, (T_i)_{i\in[\ell]}, \mathsf{ct}, \pi) :$

- *Parse* $\mathsf{ct} = (h, c, c_0, (c_j)_{j\in[n]}, (c'_i, a_i, t_i)_{i\in[\ell]})$
- *Parse* $\pi = (\mathsf{C}, \pi_1, \pi_2, \pi_3)$.
- *Set* $h_{T,i} = e(t_i, g_2)$ *for* $i \in [\ell]$
- *Output 1, if* $\mathsf{V}_1(\mathsf{ct}, (\mathsf{vk}_j)_{j\in[n]}, \pi_1) = 1$, $\mathsf{V}_3(\mathsf{CRS}, \mathsf{C}, \pi_3) = 1$, *and* $\mathsf{V}_2(((h, (h_{T,i})_{i\in[\ell]}), \mathsf{CRS}, (c, c_0, (c'_i)_{i\in[\ell]}), \mathsf{C}), \pi_2) = 1$. *Otherwise, output 0.*

Theorem 17. *Under the BDH assumption,* $\mathsf{SWE.Prove}, \mathsf{SWE.Vrfy}$ *extend* SWE *to be a verifiable SWE as defined in Def. 70. The security proof is given in the ROM.*

Proof. Given the BDH assumption, we can assume that the underlying Pedersen commitment is computationally binding, our commitment com is perfectly binding and $(\mathsf{P}_1, \mathsf{V}_1)$,$(\mathsf{P}_2, \mathsf{V}_2)$,$(\mathsf{P}_3, \mathsf{V}_3)$ are NIZK/NIZKPoK proof systems as outlined above. Specifically, Pedersen commitments and Bulletproof-instantiated $(\mathsf{P}_3, \mathsf{V}_3)$ rely on the discrete logarithm holding in the underlying group $\mathbb{G}$. For simplicity we assumed that $\mathbb{G} = \mathbb{G}_1$ and in this case, in order for BDH to hold in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ it is necessary for the discrete log to be hard in $\mathbb{G}_1$. The other assumptions are already shown in this chapter in the ROM under the BDH assumption.

Since we only combine three NIZK proofs $(\mathsf{P}_1, \mathsf{V}_1), (\mathsf{P}_2, \mathsf{V}_2), (\mathsf{P}_3, \mathsf{V}_3)$, we can establish completeness, soundness and zero-knowledge of the resulting proof system somewhat routinely.

Completeness. Given $(V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n), (T_i)_{i\in[\ell]}, \mathsf{ct}), ((m_i)_{i\in[\ell]}, w, r')) \in \mathcal{R}'$, CRS for COM and choosing $\boldsymbol{\rho} \leftarrow \mathbb{Z}_p^\ell$, $\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, (m_i)_{i\in[\ell]}; \boldsymbol{\rho})$, it must hold that $\mathsf{ct} = \mathsf{SWE.Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, (m_i)_{i\in[\ell]}); r')$ and $\forall i \in [\ell] : m_i \in \{0, \ldots, 2^k - 1\}$ and $(m = \sum_{i\in[\ell]} 2^{(i-1)k} m_i, w) \in \mathcal{R}$. The last two conditions combined with the choice of C in our construction are enough to invoke completeness on the proof π_3 output by P_3. Further, it clearly holds that $((\mathsf{ct}, V, (T_i)_{i\in[\ell]}), ((m_i)_{i\in[\ell]}, r') \in \mathcal{R}_1$ and $(((c, c_0, (c'_i)_{i\in[\ell]}), \mathsf{C}), ((m_i)_{i\in[\ell]}, r, r_0, \boldsymbol{\rho})) \in \mathcal{R}_2^{\mathsf{crs},\mathsf{CRS}}$ with $\mathsf{crs} = (h, (h_{T,i})_{i\in[\ell]})$ and $c, c_0, (c'_i)_i, h, (h_{T,i})_i$ all taken from ct and r, r_0 taken from r'.

Now, we can use completeness of the underlying proof systems and can be certain that all proofs π_1, π_2, π_3 created from $\mathsf{SWE.Prove}$ are in fact such that the calls to $\mathsf{V}_1, \mathsf{V}_2, \mathsf{V}_3$ in $\mathsf{SWE.Vrfy}$ will return 1.

Zero-Knowledge. The proof that we output consists of $(\mathsf{C}, \pi_1, \pi_2, \pi_3)$. Recall that Pedersen commitments are perfectly hiding. Specifically, if we draw $\mathsf{C} \leftarrow\!\!\$\ \mathbb{G}^\ell$, then for *any* $\boldsymbol{m}$ there exists a corresponding $\boldsymbol{\rho}$ which makes $\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, \boldsymbol{m}; \boldsymbol{\rho})$ true. So, our simulator simply samples C at random and then invokes the simulators due to the zero-knowledge property of the underlying proof systems, to simulate the π_i without access to a witness. C is clearly identically distributed to the honest $\mathsf{SWE.Prove}$ algorithm and so are the π_i by the zero-knowledge property of the underlying proof systems.

Soundness. We consider CRS to be fixed. Towards contradiction, let us assume an adversary $\mathcal{A}$ against the soundness experiment, which succeeds with non-negligible probability. When it wins, it gives an input statement $\mathsf{stmt} = (V = (\mathsf{vk}_j)_{j\in[n]}, (T_i)_{i\in[\ell]}, \mathsf{ct})$ and a proof $(\mathsf{C}, \pi_1, \pi_2, \pi_3)$ which verify in $\mathsf{SWE.Vrfy}$, but for which no valid witness $\mathsf{wit} = ((m_i)_{i\in[\ell]}, w, r')$ exists, such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}'$.

By soundness of $\mathsf{P}_1, \mathsf{V}_1$, we can establish that (except with negligible probability), $\mathsf{ct} = \mathsf{SWE.Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, \boldsymbol{m} = (m_i)_{i\in[\ell]}); r')$ for some randomness r' and some message vector $\boldsymbol{m}$.

By weak simulation extractability of $\mathsf{P}_2, \mathsf{V}_2$ and non-negligible advantage of $\mathcal{A}$, we can establish that with non-negligible probability, we can run an extractor, which may need to rewind $\mathcal{A}$ and which gives us $\bar{\boldsymbol{m}} \in \mathbb{Z}_p^\ell$ and randomnesses $r, r_0 \in \mathbb{Z}_p$, $\boldsymbol{\rho} \in \mathbb{Z}_p^\ell$ such that $(c, c_0, (c'_i)_{i\in[\ell]}) = \mathsf{com.Commit}(\mathsf{crs}, \bar{\boldsymbol{m}}; r, r_0)$, $\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, \bar{\boldsymbol{m}}; \boldsymbol{\rho})$.

Recall, the commitment com is defined by $\mathsf{com.Commit}(\mathsf{crs}, \bar{\boldsymbol{m}}; r, r_0) = (c = g_2^r, c_0 = h^r \cdot g_2^{r_0}, (c'_i = h_{T,i}^{r_0} \cdot g_T^{\bar{m}_i})_i)$ and $\mathsf{crs} = (h, (h_{T,i})_i = e(t_i, g_2)_i)$. By construction, $c, c_0, (c'_i)_i, h, (t_i)_i = H(T_i)_i^{\alpha_i}$ must be identical to the values in ct.

We discussed above, that $(h, c, c_0, (c'_i, h_{T,i})_i)$ which is fixed here in ct already constitutes a perfectly binding commitment to $\boldsymbol{m}$. So, we learn, that $\boldsymbol{m} = \bar{\boldsymbol{m}}$ must hold.

That is, we have so far shown, that there is some $\boldsymbol{m}, r', \boldsymbol{\rho}$ such that

$$\mathsf{ct} = \mathsf{SWE.Enc}(1^\lambda, V, (T_i)_{i\in[\ell]}, \boldsymbol{m} = (m_i)_{i\in[\ell]}); r') \text{ and}$$
$$\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, \boldsymbol{m}; \boldsymbol{\rho})$$

.

Now, again by weak simulation extractability of $\mathsf{P}_3, \mathsf{V}_3$ and non-negligible advantage of $\mathcal{A}$, we can establish that with non-negligible probability, we can again

successfully run an extractor, which gets full access to $\mathcal{A}$ and outputs $\boldsymbol{m}', \boldsymbol{\rho}'$ such that $\mathsf{C} = \mathsf{COM.Commit}(\mathsf{CRS}, \boldsymbol{m}'; \boldsymbol{\rho}')$ and $m_i \in \{0, \ldots, 2^k - 1\}$ for all $i \in [\ell]$ as well as $(\sum_{i\in[\ell]} 2^{(i-1)k} m'_i, w) \in \mathcal{R}$.

In case $\boldsymbol{m} = \boldsymbol{m}'$, $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}'$ must actually already hold. But, if $\boldsymbol{m} \neq \boldsymbol{m}'$, we can use $\boldsymbol{\rho}, \boldsymbol{\rho}'$ to break the computational binding of the underlying Pedersen commitment scheme COM. Thus, except with negligible probability, $\boldsymbol{m} = \boldsymbol{m}'$ must hold.

But then, we have actually shown, that the probability of $\mathcal{A}$ winning in the soundness experiment must have been negligible. This concludes our proof. □

4.7 McFly Protocol

In this section, we describe how to build a general-purpose time-release encryption mechanism, that we call McFly, by integrating a verifiable signature-based witness encryption SWE with a blockchain. The time-release mechanism is available to all users of the underlying blockchain. This allows for countless complex applications with a minimal overhead. First we introduce a formal model for the underlying blockchain and then we give a concrete instantiation of the McFly protocol. Later, we discuss concrete applications that can take advantage of the time-release encryption mechanism provided by the combination of SWE and the blockchain.

4.7.1 Formal Model and Guarantees

In this section we introduce a simplified model for blockchains in the form of the $\mathcal{BC}_{\lambda,H}$ functionality reflecting the requirements introduced in Section 4.1.

As we are modelling BFT blockchains and blockchains coupled with a finality layer, all the blocks in our abstraction are final and cannot be rolled-back. Parties only require the (multi-)signatures of the committee members on the block counter to decrypt ciphertexts, thus the blockchain in our model simply consists of a list T containing these signatures. On every tick, committee members sign the block and the new counter *ctr* as the block header. Our model takes a security parameter λ and two hash functions H, H_{pr} as parameters.

Let the number of committee members be n and the corruption threshold of the adversary be $c < n/2$. We allow the adversary to corrupt up to c parties statically. That is, the adversary may choose their signing keys in the beginning of the execution and input the signatures used in making the multi-signatures on the block counter for these parties later on.[6] We additionally require an online-extractable proof of knowledge for the public keys of committee members. Further, the adversary gets to decide when to make a new block (up to delay Δ_τ per round) by calling Tick – we allow them full control over the content of these blocks, except for the fixed block counters. They also get to see all unaggregated signatures by the honest parties.

[6] We consider static corruptions for simplicity. However, we can easily extend our model to allow for *delayed* active corruptions. In such a model, the committees are dynamically sampled and the adversary is only allowed to corrupt parties after a specified delay; this delay can then be set to be larger than the time a new committee is sampled.

Functionality $\mathcal{BC}_{\lambda,H}$

Initialization

$\mathsf{T} := ()$
$ctr := 0$ ▷ Current number of blocks
$\mathcal{C} := \emptyset$ ▷ Set of corrupted parties
Wait until $(\mathsf{Corrupt}, \cdot)$ is called by adversary $\mathcal{A}$
for $i \in [n]$ **do**
  if $i \notin \mathcal{C}$ **then**
    Sample $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{Sig_{agg}.KeyGen}(1^\lambda)$
    Output $\pi_i \leftarrow \mathsf{Sig_{agg}.Prove}(\mathsf{vk}_i, \mathsf{sk}_i)$ to $\mathcal{A}$.
$\mathcal{V} := \{\mathsf{vk}_i\}_{i\in[n]}$

Public Interface

Input: $(\mathsf{QueryAt}, CTR)$
  if $CTR \leq \mathsf{T.len}()$ **then**
    return $\mathsf{T}[CTR]$

Input: $(\mathsf{QueryTime})$
  return $\mathsf{T.len}()$

Input: $(\mathsf{QueryKeys})$
  return $\mathcal{V}$

Interface for Adversary $\mathcal{A}$

Input: $(\mathsf{Corrupt}, C', \{\mathsf{vk}'_i\}_{i\in C'}, \{\pi_i\}_{i\in C'})$ ▷ This can be called once during initialization, modelling static corruption
  if $|C'| \leq c$ and $C' \subset [n]$ and $\mathsf{Sig_{agg}.Valid}^{H_{pr}}(\mathsf{vk}'_i, \pi_i)$ for $i \in C'$ **then**
    $\mathcal{C} := C'$
    for $i \in \mathcal{C}$ **do**
      Set $\mathsf{vk}_i = \mathsf{vk}'_i$
Input: (Tick, B)
  $ctr+ = 1$
  for $i \in ([n] \setminus \mathcal{C})$ **do**
    Output $(\sigma_i) \leftarrow \mathsf{Sig_{agg}.Sign}(\mathsf{sk}_i, H(ctr))$ to $\mathcal{A}$
    Output $(\sigma'_i) \leftarrow \mathsf{Sig_{agg}.Sign}(\mathsf{sk}_i, H(ctr, B))$ to $\mathcal{A}$
  Set $S := [n] \setminus \mathcal{C}$.
  Await **input** $(C', (\sigma_i)_{i\in C'})$ from $\mathcal{A}$
  if for all $i \in C', \mathsf{Sig_{agg}.Vrfy}(\mathsf{vk}_i, H(ctr), \sigma_i) = 1$ **then**
    $S := S \cup C'$.
  Append $(\mathsf{Sig_{agg}.Agg}((\sigma_i)_{i\in S}, (\mathsf{vk}_i)_{i\in S}), (\mathsf{vk}_i)_{i\in S})$ to T.

Restrictions on the Adversary

For each round r_i, the adversary is required to send a message (Tick, B) for some block content B within time Δ_τ.

Protocol Guarantees. Let $\mathcal{L}_0$ be an NP language defined by relation $\mathcal{R}_0$ via $m \in \mathcal{L}_0 \Leftrightarrow \exists w$ s.t. $(m, w) \in \mathcal{R}_0$. Our protocol McFly consists of five algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Vrfy})$ in a hybrid model where access to the public interface

of the functionality $\mathcal{BC} = \mathcal{BC}_{\lambda,H}$ is assumed with committee size $n = \mathsf{poly}(\lambda)$ and corruption threshold $c < n/2$. The syntax of these algorithms is as follows:

$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$: Setup takes a security parameter λ. It outputs a common reference string CRS.

$\mathsf{ct} \leftarrow \mathsf{Enc}^{\mathcal{BC}}(1^\lambda, m, d)$: Encryption takes a security parameter λ, a message m and an encryption depth d. It outputs a ciphertext ct.

$m \leftarrow \mathsf{Dec}^{\mathcal{BC}}(\mathsf{ct}, d)$: Decryption takes a ciphertext ct and an encryption depth d. It outputs a message m.

$\pi \leftarrow \mathsf{Prove}^{\mathcal{BC}}(1^\lambda, \mathsf{CRS}, \mathsf{ct}, m, d, w_0, r)$: The proving algorithm takes a security parameter λ, CRS, a message m, an encryption depth d, a witness w_0 and randomness r. It outputs a proof π.

$b \leftarrow \mathsf{Vrfy}^{\mathcal{BC}}(\mathsf{CRS}, \mathsf{ct}, \pi, d)$: The verification algorithm takes CRS, a ciphertext ct, a proof π and an encryption depth d. It outputs a bit b.

We prove the following security guarantees for McFly, which are inspired by traditional time-lock puzzles:

Definition 71 (Correctness)**.** *A protocol* $\mathsf{McFly} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Vrfy})$ *is correct, if for any parameter* λ*, message* m*, depth* d*, and algorithm* $\mathcal{A}$ *running the adversarial interface in* $\mathcal{BC}$*, if* $\mathsf{ct} \leftarrow \mathsf{Enc}^{\mathcal{BC}}(1^\lambda, m, d)$ *is run at any point and* $\mathsf{McFly.Dec}^{\mathcal{BC}}(\mathsf{ct}, d)$ *is run, when the number of finalized blocks* $\mathcal{BC}.\mathsf{QueryTime}$ *is at least* d*, it will output* m*, except with negligible probability.*

Definition 72 (Security)**.** *A protocol* $\mathsf{McFly} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Prove}, \mathsf{Vrfy}, \mathsf{Dec})$ *is secure, if for any parameter* λ *and committee size* $n = \mathsf{poly}(\lambda)$*, corruption threshold* $c < n/2$ *there is no PPT adversary* $\mathcal{A}$ *with more than negligible advantage* $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Lock}} = |\Pr[b = b'] - \frac{1}{2}|$ *in the experiment* $\mathsf{Exp}_{\mathsf{Lock}}(\mathcal{A}, 1^\lambda)$ *defined below:*

1. *The experiment computes* $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$ *and outputs it to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *gets to use the adversarial interface in* $\mathcal{BC}$*, which is run by the experiment.*
3. *At some point,* $\mathcal{A}$ *sends two challenge messages* m_0, m_1 *and a depth* $d > 0$*.* $|m_0| = |m_1|$ *must hold.*
4. *The experiment draws* $b \leftarrow\!\!\$\ \{0,1\}$*.*
5. *Run* $\mathsf{ct} \leftarrow \mathsf{Enc}^{\mathcal{BC}}(1^\lambda, m_b, d)$ *and send* ct *to* $\mathcal{A}$*.*
6. $\mathcal{A}$ *can submit a bit* b' *at any point while* $ctr < d$ *in* $\mathcal{BC}$
7. *Once* $ctr \geq d$ *on* $\mathcal{BC}$ *with no prior input from* $\mathcal{A}$*,* $b' \leftarrow\!\!\$\ \{0,1\}$ *is set instead.*

Remark. There are two different points in time that are used in these guarantees: When *ctr* is incremented, this can be seen as the point in time when it is clear that the committee has reached agreement, and that a new finalized block will be added. When the multi-signature is added to T symbolizes the point in time when the finalized block (including the committee signatures) becomes available to all honest users on the blockchain (even outside of the committee). This means that there is a small gap in security: We point out that our synchronous network model guarantees that all honest parties receive the messages of other honest parties by the end of each round. However, the best practical guarantee that we can hope to achieve in terms of security is, that an adversary corrupting up to $n/2 - 1$ committee members is unable to decrypt a ciphertext before seeing any honest member's signature. However, this might occur before a block is made available to honest users. In practice, such a gap exists naturally, as we also need to account for communication and network delay.

We additionally require a verifiability property:

Definition 73 (Verifiability). *A protocol* $\mathsf{McFly} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Vrfy})$ *is verifiable for an NP language* $\mathcal{L}_0$ *with witness relation* $\mathcal{R}_0$*, if* $(\mathsf{Prove}, \mathsf{Vrfy})$ *is a NIZK proof system for a language* $\mathcal{L}'$ *given by the following induced relation* $\mathcal{R}'$*:*

$$(V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n), d, \mathsf{ct}), (m, r, w_0)) \in \mathcal{R}' \Leftrightarrow$$
$$\mathsf{ct} = \mathsf{McFly.Enc}(1^\lambda, m, d; r, V) \wedge (m, w_0) \in \mathcal{R}_0.$$

Here, $\mathsf{Enc}(\ldots; r, V)$ denotes, that the randomness used is r and the keys obtained from the blockchain are V.

Note that this guarantees that 1) a receiver of a verifying pair (ct, π) can be sure to retrieve an output in $\mathcal{L}_0$ after block d was made and 2) outputting π alongside ct reveals no further information.

4.7.2 Protocol Description

We now describe how to build McFly based on an SWE scheme and a blockchain functionality $\mathcal{BC}_{\lambda,H}$ as described above.

Construction 8. *Let*

- SWE *be a verifiable SWE*
- $\mathsf{COM} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Vrfy})$ *be a Pedersen commitment*
- H *be the hash function in* $\mathcal{BC}$ *and* H_2 *be another hash function.*

H, H_2 are implicitly made available in all calls to SWE*, which is set up for parameters $t = n/2$ out of n. k is the upper bound on the message lengths for* SWE.[7] *We now describe* McFly*:*

$\mathsf{Setup}(1^\lambda)$:

- *Return* $\mathsf{COM.Setup}(1^\lambda)$.

$\mathsf{McFly.Enc}^{\mathcal{BC}}(1^\lambda, m, d)$:

- *Get the keys V by calling* QueryKeys *to* $\mathcal{BC}$.
- *Split* $m = (m_i)_{i\in[\ell]}$ *where* m_i *are from* $\{0, \ldots, 2^k - 1\}$ *with* $m = \sum_{i\in[\ell]} 2^{(i-1)k} m_i$.
- $\mathsf{ct} \leftarrow \mathsf{SWE.Enc}(1^\lambda, V, (H(d))_{i\in[\ell]}, (m_i)_{i\in[\ell]})$.
- *Output* ct.

$\mathsf{McFly.Dec}^{\mathcal{BC}}(\mathsf{ct}, d)$:

- *If* QueryTime *returns less than d, abort.*
- *Get (σ, U) by calling* $(\mathsf{QueryAt}, d)$ *and V by calling* QueryKeys *to* $\mathcal{BC}$.
- *Call* $(m_i)_{i\in\ell} \leftarrow \mathsf{SWE.Dec}(\mathsf{ct}, (\sigma)_{i\in[\ell]}, U, V)$.
- *Output* $m = \sum_{i\in[\ell]} 2^{(i-1)k} m_i$.

$\mathsf{McFly.Prove}^{\mathcal{BC}}(1^\lambda, \mathsf{CRS}, \mathsf{ct}, m, d, w_0, r)$:

- *Get the keys V by calling* QueryKeys *to* $\mathcal{BC}$.
- *Split* $m = (m_i)_{i\in[\ell]}$ *s.t.* $m = \sum_{i\in[\ell]} 2^{(i-1)k} m_i$.
- *Output* $\pi \leftarrow \mathsf{SWE.Prove}(\mathsf{CRS}, V, (H(d))_{i\in[\ell]}, \mathsf{ct}, (m_i)_{i\in[\ell]}, w_0, r)$.

$\mathsf{McFly.Vrfy}^{\mathcal{BC}}(\mathsf{CRS}, \mathsf{ct}, \pi, d)$:

- *Get the keys V by calling* QueryKeys *to* $\mathcal{BC}$.
- *Output* $b \leftarrow \mathsf{SWE.Vrfy}(\mathsf{CRS}, V, (H(d))_{i\in[\ell]}, \mathsf{ct}, \pi)$

[7] The size limit is required for efficient decryption, see § 4.3.2.

4.7.3 Proofs

In this section we show that the McFly protocol satisfies the definitions of § 4.7.1.

Theorem 18. McFly *is correct, given that* SWE *has robust correctness.*

Proof. Let a parameter λ, a message m^*, a depth d and an algorithm $\mathcal{A}$ be given. Let $\mathsf{ct} \leftarrow \mathsf{Enc}^{\mathcal{BC}}(1^\lambda, m^*, d)$.

We show that if $\mathsf{McFly}^{\mathcal{BC}}.\mathsf{Dec}(\mathsf{ct}, d)$ is run, when the number of finalized blocks $\mathcal{BC}.\mathsf{QueryTime}$ is greater or equal d, it outputs m. By construction, we have $\mathsf{ct} \leftarrow \mathsf{SWE.Enc}(1^\lambda, V, (H(d))_{i\in[\ell]}, (m_i^*)_{i\in[\ell]})$, where V is the (static) set of keys obtained from $\mathcal{BC}$ and $(m_i^*)_{i\in[\ell]}$ is the result of splitting m^* into chunks from $\{0,\dots,2^k-1\}$.

In our call to Dec, since $\mathcal{BC}.\mathsf{QueryTime}$ is greater or equal d, we can call $(\mathsf{QueryAt}, d)$ and receive (σ, U), which is by definition, such that

$$(\sigma, U) = (\mathsf{Sig_{agg}.Agg}((\sigma_i)_{i\in S}, (\mathsf{vk}_i)_{i\in S}), (\mathsf{vk}_i)_{i\in S}),$$

where for all $i \in [S]$ $\mathsf{Sig_{agg}.Vrfy}(\mathsf{vk}_i, H(d), \sigma_i) = 1$. By correctness of $\mathsf{Sig_{agg}}$, it holds $\mathsf{Sig_{agg}.AggVrfy}(\sigma, U, H(d)) = 1$. We then call $m \leftarrow \mathsf{SWE.Dec}(\mathsf{ct}, (\sigma)_{i\in[\ell]}, U, V)$.

Now, if there was an index ind such that $m_{\mathsf{ind}} \neq m_{\mathsf{ind}}{}^*$, forwarding that ind, V, U, $(m_i)_i$, $(T_i)_i$ and σ to the experiment for robust correctness of SWE would constitute a winning adversary. Thus, except with negligible probability $m = m^*$, concluding the proof.

□

Theorem 19. McFly *is secure given that* SWE *is secure and* H *is collision resistant.*

Proof. Let λ, committee size $n = \mathsf{poly}(\lambda)$ and a corruption threshold $c < n/2$ be given. Let us assume towards contradiction, that there is an adversary $\mathcal{A}$ with non-negligible advantage ε in $\mathsf{Exp}_{\mathsf{Lock}}(\mathcal{A}, 1^\lambda)$.

First, we discuss a hybrid game $\mathcal{H}_1$: It corresponds to the real experiment, but once we received d from $\mathcal{A}$, if $\mathcal{A}$ has received a signature on $H(d)$ before, we abort. If they query on a signature on that message afterwards, we also abort. If $\mathcal{A}$ had a non-negligible advantage in differentiating $\mathcal{H}_1$ from the experiment, they would have to have a non-negligible advantage in causing an abort. If this were the case, we could directly build a reduction against collision resistance of H.

Thus, we now assume we have an adversary who wins in $\mathcal{H}_1$ with non-negligible probability. We describe a reduction to security of SWE for $t = n/2$ out of n for the set of indices $S = [n]$ at which the challenge message is included. W.l.o.g. we assume n is even.

- The reduction gets access to H_{pr} from the experiment and sets up $\mathcal{BC}$ with access to H_{pr}.
- It computes $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$ and outputs it to $\mathcal{A}$.
- It honestly simulates $\mathcal{BC}$ to $\mathcal{A}$, except in the way it generates the keys and answers signing queries.
- In the initialization:
  - Let C' be the indices of malicious keys chosen by $\mathcal{A}$ and set $\bar{C} = [n] \setminus C'$. Let $V' = (\mathsf{vk}_i)_{i\in C'}$ be the malicious keys and π_i the corresponding proofs.
  - The reduction receives $n/2+1$ keys $V_E = \mathsf{vk}_i'$ from the experiment and sets the first $n/2+1$ of the honest keys $(\mathsf{vk}_i)_{i\in\bar{C}}$, to be these vk_i'. If the adversary chose $|C'| < n/2-1$, the remaining honest keys are generated by $\mathsf{Sig_{agg}.KeyGen}$ and saved as V_R. The proofs of validity for keys in $\mathcal{V}_E$ are received from the experiment and for $\mathcal{V}_R$, the reduction computes them honestly.

- For signing:
  - For keys in V_R we sign honestly.
  - For keys in V_E we relay signing queries to the experiment.
- Then, we output $V_R \cup V'$ as challenge keys to the experiment, where we receive the validity proofs for V' from $\mathcal{A}$. These verify by definition of $\mathcal{BC}$.
- We receive messages m_0, m_1 and a depth $d > 0$ from $\mathcal{A}$. We choose $(m_i^0)_i \in [\ell]$ as a split of m_0 and $(m_i^1)_i \in [\ell]$ as a split of m_1. We output $(m_i)_i$ and $T_i = d$ for all $i \in [\ell]$ to the experiment.
- We receive back a ciphertext ct that we forward to $\mathcal{A}$.
- Now, if $\mathcal{A}$ outputs a bit b in time, we output it to the experiment. Otherwise, we output a random bit.

Note, that by our abort conditions, we have not asked for a signature on T_{ind} before outputting it nor afterwards, causing the interaction with the experiment to run through.

If the experiment chooses bit b', our output is identically distributed to running $\mathcal{H}_1$ with $b = b'$. Since we also guess randomly, if $\mathcal{A}$ does not output a bit, that means, that we get the same advantage as $\mathcal{A}$ does in $\mathcal{H}_1$. Assuming security of SWE, that concludes the proof.

□

Theorem 20. McFly *is verifiable, given that* SWE *is a verifiable SWE.*

This follows immediately by the definition of McFly.Enc and verifiability of SWE, as we only invoke the underlying NIZK proof system.

4.7.4 Integration with Casper

As a concrete example, we discuss the modifications necessary to the Ethereum 2.0 Altair [Tra] protocol running with the Casper finality layer [BG19]. In the Casper protocol [BG19] the committees are randomly sampled and become responsible for finalizing blocks for some period of time, called an epoch. The members of the committee can cast votes on the blocks they believe to be final by signing the blocks using their signing keys. Once a majority of the committee votes on the same block it becomes final. As new committees are chosen only one epoch in advance and each epoch lasts for about 6.4 minutes, we can at most encrypt to 12.8 minutes into the future. Moreover, a final block is only produced roughly after every epoch duration. So while we have a near-constant block production rate, the horizon choice is essentially limited to the start of the next epoch. However, an extension of the above is possible in the current version of Ethereum Altair [Tra; But]. There, the so-called sync committees have a lifetime of roughly 27 hours and are appointed "one round" in advance.[8] These committees periodically sign the header of the newest block to enable faster verification for light clients. We believe that a horizon of 27 hours is sufficient for most applications, including the ones we will describe in § 4.7.5. We stress however that the security implications for the concept of sync committees were not formally analyzed in Ethereum Altair [Tra; But].

We describe the modifications needed on Ethereum 2.0 in order to instantiate McFly on top of it:

- Since Casper already uses BLS signatures, there are two possible alternatives. 1) The committee of Casper adopts the aggregation mechanism described in § 4.4, or 2) decryptors obtain the unaggregated signatures themselves (which is

[8] When a new sync committee becomes active, the next one is sampled.

already possible with the Ethereum beacon chain).[9] In the latter case, signature aggregation can be performed locally at the decryptor and hence no modifications are necessary. One could also envision dedicated aggregation servers when the system is widely adopted.
- For each finalized block the (sync) committee additionally signs a counter r that represents the number of finalized blocks (or slot number).
- The public keys of the committee members must have a proof of knowledge. This can be achieved, e.g., by registering the keys with a PKI.

Extension for Dynamic Committees. In our model, we assumed static committees. However, a fixed committee can be a security risk for a finality layer deployed in the wild as committee members usually become target of attacks. Hence, finality layers advocate for a short-lived dynamic committee [DMMNT20], mitigating this risk.

We can safely regard a committee as known and static during its lifetime. Thus, our model naturally extends as long as we only encrypt messages as far into the future as the committees are currently known.

Moreover, in the period where the current committee is about to be replaced and the next committee is not yet known, the limitation becomes more severe. However, this can be mitigated by considering interleaved committees for the finality layer; one committee starts to be active at the middle of the lifetime of the other, and both committees independently run the protocol. This ensures a smooth transition between committees.

4.7.5 Applications

In this section we describe some applications of our McFly protocol.

Encrypted Mempool to Prevent Frontrunning. Blockchains are vulnerable to *frontrunning*: once a plaintext transaction (e.g., a profitable trade) appears in the mempool, the public pool of transactions pending inclusion, adversaries scan it and craft counter-transactions that target the same opportunity while offering higher fees, aiming to be included *before* the original. In some settings, miners/validators may even reorder transactions directly. In all cases, the attack is enabled by *early content visibility* that makes profitable reordering possible.

Prior work proposed keeping transactions hidden in the mempool using time-lock primitives such as time-lock puzzles and verifiable delay functions [DDMMT20; BBBF18]. This approach does not prevent reordering per se, but removes the content signal that underpins classic mempool-based frontrunning. However, such time-release mechanisms are resource-intensive and brittle to parameterization across heterogeneous hardware. Usually, a reference hardware is used to set parameters for time-lock puzzles to ensure the data stays hidden even from the fastest machines, however in decentralized systems like blockchain some hardware may have a performance gap of up to 1000 times slower computation speed than reference hardware. Our efficient McFly protocol provides a drop-in alternative for time-release encryption in this scenario, where encryption in terms of block-height is a very natural setting.

Decentralized Auctions. An auction is a process of trading goods for bids, and it is run among an arbitrary number of bidders, and an auctioneer. First, in a bidding phase, all the bidders submit their bids. Then, in a final phase the auctioneer, after

[9] See https://github.com/ethereum/consensus-specs/blob/dev/specs/phase0/p2p-interface.md#attestation-subnets

checking all bids announces the winner. If the auctioneer is not fully honest, it is easy to see that the outcome of the auction cannot be trusted. Therefore, protocols for decentralized auctions (or auctioneer-free protocols) are of great interest. Here we focus on sealed-bid auctions that are run exclusively on a blockchain, i.e., the communication of the parties happens only through blockchain transactions. Additionally, the set of parties running the blockchain and bidding on the auction can overlap, making it profitable for malicious parties to, e.g., not include bids that are greater than their own.

One difficulty in realizing decentralized auctions in blockchains, as remarked by Deuber et al. [DDMMT20], is that a bid transaction should be self-contained, i.e., the auction protocol should be able to terminate and determine the correct winner after collecting all bid transactions. Particularly, this rules out solutions where the bid transaction is a commitment to the real bid, and later an opening to this commitment is sent. The reason is that if no opening is ever received, it is not possible to determine if the opening was suppressed by malicious parties in the blockchain or a malicious bidder is refusing to open his bid to DoS the auction. This is handled in [DDMMT20] by including in the bidding transaction a time-lock puzzle containing the opening of the commitment to the bid; whenever an opening is not received, the auction can still terminate by solving the time-lock puzzles and retrieving the bids. This comes with all the disavantages of timelock puzzles discussed in the introduction.

In contrast, with the proposed time-release encryption mechanism, we can easily realize the same decentralized auction as [DDMMT20] without any overhead on the blockchain, by simply encrypting the bids for a future block. We give next a high level description of this. Assume we want the bidding phase to start at final block number r in the underlying blockchain, and span through a sequence of ℓ final blocks. During the bidding phase, all parties will encrypt their bids with respect to the committee members' verification keys and to the counter $r+\ell$. After this final block is signed by the committee, the signature can be used to decrypt all the bids, making it possible to publicly verify who the auction winner is. Note that with this simple approach there is no bid privacy for the losing bids, as all the bids are opened at the end of the auction run. Moreover, if the underlying blockchain is incentive-compatible, the results of [DDMMT20] show that the encrypted bids will always be included in the blockchain given that it pays the required fees.

Randomness Beacons. A randomness beacon is a tool that provides public randomness at predefined intervals. Blockchains can be a great source of randomness to build decentralized randomness beacons, where distrustful parties contribute to the randomness output by the beacon. One known problem with using blockchains for building randomness beacons is that malicious parties could somehow manipulate the contents to be included in the blockchain as a way to introduce bias in the beacon output. For that, some solutions [LW15; BBBF18; BDDNO21] leverage the use of verifiable-delay functions (VDF) to delay the output of the beacon, such that malicious parties do not have enough time at their disposal to be able to bias the randomness. However, VDFs carry the same drawbacks as time-lock puzzles; they are wasteful and it is usually hard to set parameters that result in a secure and usable system. As the role of VDFs in these randomness beacon constructions is simply to hide information from the parties for a predetermined period of time, with only minor modifications one could replace the VDFs for our time-release encryption mechanism in [LW15; BBBF18; BDDNO21] to get randomness beacons with almost no overhead on the blockchain.

Chapter 5

Compact Signature Witness Encryption in the Standard Model

In this chapter, we present an alternative construction of Signature Witness Encryption, which improves upon asymptotic complexity and resides in the standard model, avoiding idealized assumptions such as the ROM. Our construction is *compact* in the sense that ciphertext size scales polylogarithmically in the number of potential signers. This improves upon the previous linear dependency in SWE and adjacent schemes [DHMW23; Mad+23].

Furthermore, compared to our previous definition of SWE (§ 4.3.2), we only achieve a relaxed non-adaptive security definition. To capture all the model nuances, we formalize *Compact Signature Witness Encryption* (cSWE) in a new definition and compare it to our previous SWE definition.

The chapter is taken almost verbatim from [ADMSW24], which is joint work with Gennaro Avitabile, Nico Döttling, Bernardo Magri and Christos Sakkas published at Asiacrypt 2024.

5.1 Structure and Contributions

The structure of this chapter is as follows:

- In § 5.2, we provide a technical outline detailing the high-level ideas of our construction, comparisons to related work, and a discussion of our results.
- In § 5.3, we define the notion of *strongly puncturable signatures* (SPS) that we will work with, and we introduce the notion of *compact signature witness encryption* (cSWE) for SPSs with non-adaptive security. We include a short comparison with our previous notion of SWE.
- In § 5.4, we proceed to instantiate our cSWE scheme for SPSs.

We now outline our contribution in a little more detail:

Compact Signature Witness Encryption. The cSWE scheme that we introduce is focused on small asymptotic complexity and standard model instantiation, however it is not concretely efficient. We hope it can pave the way to future more efficient constructions.

It allows one to encrypt a message m with respect to a reference tag T and a set of verification keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ such that the ciphertext size only scales polylogarithmically in n.

Our construction relies on Turing machine indistinguishability obfuscation (compare § 2.14.1) and is based on *strongly puncturable signatures* (SPS). For context, this notion was introduced by Goyal, Vusirikala, and Waters [GVW19] and compared

to regular signature schemes has an alternative *punctured* key generation algorithm that on input a message m^* outputs a key pair $(\mathsf{vk}^*, \mathsf{sk}^*)$, such that under vk^* there *exists* no verifying signature for message m^*, but vk^* otherwise looks computationally indistinguishable from an honestly generated (unpunctured) public key.

Security is *non-adaptive* in the following sense: We require that the adversary chooses the reference message T and the indices of the corrupted keys ahead of time. Only after that does the adversary get $t-1$ honestly generated pairs of verification and signing keys (at the corrupted positions) and $n-t+1$ honestly chosen verification keys for the honest parties. We then give the adversary access to a signing oracle which will sign for all honest keys and all messages except the challenge tag T. Under these constraints, we ask for indistinguishability security for the signature witness encryption, i.e., the adversary should not be able to distinguish SWE encryptions with respect to $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and T of two distinct messages $m_0 \neq m_1$. A short discussion about the difficulty of achieving adaptive security can be found in § 5.2.3.

Informal Theorem 8 (Construction 9, Thms. 21 to 23)**.** *Assuming the existence of $i\mathcal{O}$ for Turing machines, puncturable PRFs, SSB hashing and symmetric key encryption, there exists, in the standard model, an SWE scheme for SPSs with ciphertext size in $\mathcal{O}(\mathsf{poly}(\log n, \lambda))$, where n is the number of involved signer keys and λ is the security parameter.*

Individual Contribution. Nico Döttling and Bernardo Magri initiated the project. When I was invited by Bernardo to join the discussions, Gennaro Avitabile and Christos Sakkas were already involved. There were many joint discussions, initially aiming at a different, more practical construction.

The idea for the protocol in its present form, relying on indistinguishability obfuscation, was developed primarily by Nico during a meeting with Bernardo and myself. Both Bernardo and I provided comments that led Nico to refine his initial proposal—specifically, I pointed out compactness issues in the original version.

The introduction was jointly written, while the technical outline in its current form was prepared by Nico and myself. Initially, we were unaware that the notion of strongly puncturable signatures had already been introduced; this was brought to our attention by a reviewer. The formalization of this primitive, as well as the write-up of its instantiation, which is present in the full version [ADMSW24], was handled primarily by Christos and Gennaro.

The instantiation of compact SWE from $i\mathcal{O}$, including the accompanying proofs, was written mainly by me.

5.2 Technical Overview

We will now provide an outline of our constructions and techniques.

5.2.1 Developing Compact SWE

Threshold SWE Blueprint. Let us focus on the *core* of our previous SWE scheme, forgetting about efficiency boosting strategies and multi-signature support. What we need to do in order to achieve plain t-out-of-n threshold SWE for BLS signatures is remarkably simple:

First, we notice the the Boneh-Franklin IBE directly implies a 1-out-of-1 SWE scheme for the BLS signature. For a message m, and verification key vk, we get the ciphertext $\mathsf{ct} = (g_2^r, e(H(T), \mathsf{vk})^r \cdot \langle m \rangle)$, where $\langle m \rangle$ is some representation of m in the target group $\mathbb{G}_T$. Notice, that we previously encoded m as g_T^m, but this

specific representation is only strictly necessary to support our advanced properties like verification.

Then, to achieve t-out-of-n threshold encryption, we simply share the message m using a t-out-of-n secret sharing and provide an individual ciphertext of that shape for each share.

To facilitate hybrid encryption of polynomially long messages, we can encrypt a symmetric key K in place of the message m.

This blueprint of secret sharing a message/randomness/a key and each secret share resulting in its own ciphertext part is very natural and in fact followed in essence both by our previous construction and the alternative construction of *verifiable witness encryption based on threshold signatures* presented concurrently by Madathil et al. [Mad+23].

Linear size in the signer set is inherent in this approach due to the generation of fresh ciphertext parts for each share. Looking ahead, the way that we try to salvage this will be by *compressing multiple shares into one ciphertext.*

Model and Security Considerations. We remind the reader that we operate in the standard model and consider a non-adaptive security model, where the indices to be corrupted and the challenge reference message need to be provided by the adversary up front.

While we would prefer to achieve *adaptive fully malicious security*, where the adversary can corrupt any unqualified set of signers and even choose their verification keys in a fully malicious manner, this seems to be difficult to achieve in conjunction with compactness. In § 5.2.3, we provide some considerations about this difficulty.

A First Attempt. In order to achieve a compact SWE construction without trusted setup, our basic idea is to adapt the construction of witness encryption from $i\mathcal{O}$ to the setting of compact SWE, but this raises several challenges. Hence, consider the following attempt to construct a cSWE. The ciphertext consists of a symmetric key encryption of the message m under key K and an obfuscated circuit C. The circuit C pseudorandomly generates the secret shares under a secret sharing scheme (e.g. Shamir's scheme) of key K on demand; on input an index i and a valid signature σ for T under vk_i, the circuit produces the i-th share of the key K. The randomness for generating shares is taken from a puncturable PRF.

There are two evident issues with this approach. 1) A minor issue is that we cannot hardwire all the verification keys vk_i into the circuit C, as this would require the circuit size growing linearly with n, contradicting our compactness requirement. 2) Computing individual Shamir shares inside the circuit C implies evaluating a degree $t-1$ polynomial (that takes time $\Theta(n)$), requiring a circuit of size $\Omega(n)$, again contradicting our compactness requirement.

There will be, however a more subtle third issue which surfaces when trying to prove security of this construction: If we were to rely on standard puncturing techniques to *erase* the shares of honest parties in the challenge ciphertext, we would need to puncture the above circuit $n - t$ times. But this would again require an obfuscated circuit of size $\Omega(n)$!

Our Basic Approach. Starting from the above sketch, we will address issues 1) and 2) above as follows. To address the issue 1), we will rely on somewhere statistically binding (SSB) hashing [HW15], in the same way as in our URS constructions. We recall: An SSB hash function is a keyed commitment scheme, which allows to succinctly produce a committing hash h of a database of size n. The key generation

algorithm takes an index $i \in [n]$, and outputs a hashing key, which except with negligible probability guarantees that hashes under that key are computationally binding on all indices and perfectly binding on i. The binding index is hidden by the commitment scheme.

We use the SSB scheme to commit to all verification keys $(i, \mathsf{vk}_i)_{i\in[n]}$ and force the decryptor to input $(i, \mathsf{vk}_i, \sigma)$ and a valid opening τ for the SSB hash. This ensures, that except with negligible probability on input $(i, \mathsf{vk}, \sigma, \tau)$, vk actually is the i-th key and σ is a signature for vk_i, so outputting s_i is justified. To address issue 2), we will rely on $i\mathcal{O}$ for Turing machines [KLW15; GS18]. The size of an obfuscated Turing machine (TM) depends polynomially on its description size, but only polylogarithmically on its runtime.

Towards Proving Security. The intuition for the security proof is that we want to replace the obfuscated TM M with an equivalent TM M' which has no information about the shares or the shared key. The basic idea of the hybrid argument is as follows.

Since we are in a setting of non-adaptive security, the security reduction knows which signers will be corrupted even before their corresponding verification keys are generated. We will use a standard SSB argument to go through a sequence of hybrids, where in hybrid i we make the SSB hash statistically binding to the i-th *uncorrupted* verification key, call this key vk_i. We would like to argue that since the adversary never obtains a signature σ_i of T under vk_i, he cannot make the obfuscated TM output a share s_i of the message m_i. However, standard EUF-CMA security seems far too weak to guarantee this: In order to use indistinguishability security of $i\mathcal{O}$, we need to move into a situation where it is *impossible* (rather than computationally hard) to make the TM accept some signature σ_i for T under vk_i and output the corresponding share s_i. To address this issue, we use strongly puncturable signatures (Section 5.3.1). These allow a special key generation $\mathsf{PKeyGen}(T)$, which outputs a punctured key pair $(\mathsf{vk}, \mathsf{sk})$ at message T. No valid signature exists for message T, while the verification key remains indistinguishable from a standard non-punctured key. This type of signature was introduced under the name of all-but-one signatures by Goyal, Vusirikala, and Waters in [GVW19], where various instantiations based on different number-theoretic assumptions (e.g., RSA, pairing-based assumptions, LWE) were proposed. Equipped with this tool, the reduction can now replace all honest keys with punctured keys, punctured at T. Hence, by additionally relying on SSB hashing as explained above we can rely on $i\mathcal{O}$ security to gradually replace the TM M by a TM M' which never outputs shares s_i for indices i of honest parties.

The Final Scheme. However, this transformation does not quite suffice to establish security. Recall that the shares s_i of the message m are generated *pseudorandomly*, that is an adversary with access to a plain description of the TM M' described above could still easily recover the PRF key $\tilde{K}$ used to generate the shares and just generate them by himself. Note also that $i\mathcal{O}$ security by itself does not guarantee that $\tilde{K}$ is hidden. The standard tool commonly used to argue security in this setting is puncturing: If we were to puncture the key $\tilde{K}$ in a suitable manner, then we might be able to ensure that this punctured key does not leak information about the shares of honest parties. While there are some minor issues with this idea, such as the fact that the shares s_i are correlated and a contrived puncturing argument would be necessary, the big issue here is we would need to puncture at $n - t = \Omega(n)$ points, which once again contradicts our compactness requirement! The reason why we need to remove all honest shares simultaneously is that otherwise Shamir secret sharing offers no security. Removing one share at a time is useless, as this share could easily

be recomputed from the other shares. Hence, puncturing will not help us, and we need to depart from the puncturing approach. But where could we store shares of the message m instead? Observe that our ciphertext already includes an SSB commitment to all the verification keys vk_i, and we will precisely leverage this feature in our solution: We will augment the verification keys vk_i by some additional auxiliary information which contains a sufficiently large slot to encode a Shamir share. In the real world this auxiliary information is just a random string. But in the security reduction we can embed an encryption of the corresponding share s_i into vk_i. We then use $i\mathcal{O}$ security to switch to a machine M' that reads its shares from the input, limiting its internal computation to checking the validity of the input SSB opening and the signature. Upon success, it decrypts and outputs the share in vk_i; otherwise, it aborts. This switch between computational models is feasible because we bind the SSB hash to each index $i^* \in [n]$. This ensures that on input $(i^*, \mathsf{vk}, \sigma, \tau)$, if the obfuscated machine produces an output, then $\mathsf{vk} = \mathsf{vk}_{i^*}$ must hold. Consequently, the correct share is obtained by decrypting vk, and the output remains consistent whether the shares are computed internally or decrypted from the public keys.

While this outlines our main technical ideas, there are some additional subtle technical challenges which our construction in § 5.4 will address.

5.2.2 Related Work

For related work concerning the general notion of Signature Witness Encryption and time-release application, we refer the reader to the Related Work section (§ 4.2.3) in the previous chapter. We will focus here on compactness of previous SWE schemes and other lines of works that were released concurrently with or after our paper [DHMW23] to avoid duplicate discussions.

Signature-Based Witness Encryption. Our previous construction of signature-based witness encryption [DHMW23] as described in Chapter 4 grows linearly in the number of verification keys input to the encryption procedure. The same is true for verifiable witness encryption based on threshold signatures [Mad+23]. While the ciphertext size of these constructions is asymptotically worse than ours, their main focus is on concrete efficiency. However, both [DHMW23; Mad+23] rely on the random oracle model, whereas we focus on a compact-ciphertext construction in the standard model. Furthermore, we point out that these previous works achieve fully adaptive security notions, as opposed to our non-adaptive security as discussed in § 5.2.

There also exists a concretely efficient scheme (in the ROM) with constant-size ciphertexts [GMR23], but as discussed in § 4.2.3, it requires a distributed key generation protocol to achieve a joint public key for the committee with regard to which encryption takes place.

Threshold Encryption. We previously introduced threshold encryption schemes [DF90; Fra90], where decryption can take part only with the cooperation of a threshold number of decryption servers. As we mentioned, threshold encryption has been proposed as a tool for frontrunning prevention, e.g., in [DFL24; MGZ23], where the entire mempool of pending transactions is encrypted to be released after the next block. These constructions achieve constant ciphertext size, but require correlated key setup.

Recently, Choudhuri, Garg, Piet, and Policharla [CGPP24] identified a gap in security when using either [DFL24; MGZ23] or our SWE-based time-release encryption from [DHMW23] for mempool encryption: Specifically, since encryption is suggested

to be to a (close) specific time in the future, what happens to security if some transactions are not included in the chain yet, before they are decrypted? They formalize the notion of *pending transaction privacy* demanding that such a scenario mustn't happen in mempool encryption.

This resulted in a further line of research: In [CGPP24] a threshold-encryption-based mempool encryption with pending transaction privacy with efficient batch-decryption was proposed. This solution still requires a correlated setup of secret keys, but a server can output partial decryptions for many ciphertexts at once, reducing the communication overhead. In their scheme, the keys are generated by a trusted dealer or an expensive MPC protocol, which needs to be run once per epoch.

This is improved upon in [BLT25; BFOQ25], who drop the requirement of epochs and epoch specific setups, but maintain the need for a distributed key setup.

Opposed to this, Garg, Kolonelos, Policharla, and Wang [GKPW24] introduce the notion of silent setup for threshold encryption. This means, that the involved decryption servers are now able to choose their key pairs independently and encryptors can deterministically aggregate the public keys of their chosen decryption servers into a single succinct encryption key, which can be used to encrypt a message. Dropping correlated key creation is achieved by pushing the complexity into a highly structured CRS and "hints", which can be seen as extensions of the public keys of each party. Both the CRS and the hints are linear in the size of maximum decryption servers. For each decryption procedure direct communication with the decryption servers is still required. Their scheme achieves adaptive security in the *programmable* generic group model and part of their construction relies on generalizing SWE to linearly verifiable signature schemes.

This result is further improved upon by Bormet et al. [Bor+25], who achieve batched threshold encryption with silent setup (in the CRS model) to further boost efficiency.

Our compact SWE construction requires neither distributed key generation, nor a CRS, but it is less practically efficient than (some of) these works addressing comparable settings. Therefore, we treat our construction as a feasibility result and not as a viable option for deployment.

5.2.3 Discussion and Interpretation of our Results

While we establish feasibility of compact SWE schemes, the question remains open, whether the strong tools we employ are necessary for such a construction and whether constructions achieving better notions of security are possible. As non-compact SWE is quite simple to instantiate, it may be worthwhile to study the relation of compactifying SWE with other hard problems in cryptography to develop a deeper understanding of its complexity.

On the Use of $i\mathcal{O}$. Our construction uses $i\mathcal{O}$ for Turing Machines in a crucial way. Specifically, we use $i\mathcal{O}$ to delegate the generation of shares to the decryptor, the ciphertexts in our scheme constitute compressed versions of a pseudorandom share vector. This type of compression is currently beyond the scope of other compact delegation techniques (e.g. Laconic Function Evaluation [QWW18]), and in fact general purpose delegation schemes with even a weak amount of output compression imply $i\mathcal{O}$ [LPST16]. This indicates that a different, potentially cryptographically interesting approach might be necessary to facilitate compact SWE from more favourable assumptions.

On Adaptive Security. There seems to be a substantial barrier for achieving adaptive security for *compact* SWE.

From an information theoretic perspective, e.g. for a threshold of $t = n/2$, a ciphertext of size $o(n)$ is too small to even encode the set of corrupted parties, which requires $\Omega(n)$ bits. Hence, it seems hard to make a ciphertext behave differently for honest and corrupted keys as the ciphertext cannot "know" which keys are corrupted and which are not. Hence, somewhat expectedly guessing-based transformations from non-adaptive to adaptive security fail in this setting. If we guess the set of corrupted parties ahead of time, we need to compensate for a security loss of order $2^{\Omega(n)}$. But this means that the underlying primitives need to at least provide $\Omega(n)$ bits of security, which in turn results in a ciphertext of size $\Omega(n)$, which is not compact by our definition.

5.3 Definitions

As previously established, the model we operate in and the guarantees achieved by our compact SWE construction differ from the notion of SWE introduced in Def. 67 in the previous chapter. In this section, we provide an alternative definition of compact signature witness encryption schemes instantiated with strongly puncturable signature schemes. We also highlight the differences from the previous definition. For completeness, we begin by stating the definition of strongly puncturable signature schemes.

5.3.1 Strongly Puncturable Signatures

Strongly puncturable signature (SPS) schemes are signature schemes with additional features and were introduced in [GVW19] under the name of all-but-one signatures. In particular, such a scheme comes with a punctured key generation algorithm that, on input a message m^*, generates a pair of punctured keys $(\mathsf{vk}^*, \mathsf{sk}^*)$ at message m^*. A SPS scheme has to satisfy: 1) *puncturability*, meaning that given a punctured key $(\mathsf{vk}^*, \mathsf{sk}^*)$ w.r.t. a message m^* there *does not exist* a valid signature for m^* with respect to the key vk^*; 2) *punctured-key indistinguishability* ensuring that punctured verification keys are indistinguishable from regular verification keys, as long as no signature on m^* is requested. Additionally, an SPS has to satisfy the usual correctness property of digital signatures. In [GVW19], the authors note that punctured-key indistinguishability already implies selective unforgeability, hence we will not seperately define unforgeability for our scheme.

We call this type of signature *strongly* puncturable to remark the difference with other notions of puncturable signatures [CPW20; BSW16; JDS22] where a verifying signature for the punctured message m^* might exist, but it is infeasible to compute (given e.g. a punctured signing key).

In [GVW19], various instantiations that can be based on different number-theoretic assumptions (e.g., RSA, pairing-based assumptions, LWE) were proposed in the standard model.

Definition 74 (Strongly Puncturable Signature Scheme)**.** *A* strongly puncturable signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of the following algorithms:*

- $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*: On input the security parameter* 1^λ*, the key generation algorithm* KeyGen *returns a verification key* vk *and a signing key* sk*.*
- $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{PKeyGen}(1^\lambda, m^*)$*: On input the security parameter* 1^λ*, a message* m^* *to puncture at, the punctured key generation algorithm* PKeyGen *returns a*

verification key vk *and a signing key* sk*, that allows to sign any message except* m^**.*

- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$*: On input a signing key* sk *and a message* m*, it outputs a signature* σ*.*
- $b \leftarrow \mathsf{Vrfy}(\mathsf{vk}, \sigma, m)$*: On input a verification key* vk*, a signature* σ *and a message* m*, the verification algorithm* Vrfy *returns a bit* $b \in \{0,1\}$*.*

A strongly puncturable signature scheme should have the following properties:

Definition 75 (Correctness)**.** *For all* λ *and all messages* m *in the underlying message space*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{Sign}(\mathsf{sk}, m), m) = 1 \;\middle|\; (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\right] = 1.$$

Definition 76 (Puncturability)**.** *For all* λ *and for all messages* m^**, we require*

$$\Pr\left[\exists \sigma \text{ s.t. } \mathsf{Vrfy}(\mathsf{vk}^*, \sigma, m^*) = 1 \;\middle|\; (\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow \mathsf{PKeyGen}(1^\lambda, m^*)\right] = 0.$$

Definition 77 (Punctured-Key Indistinguishability)**.** *We say that a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *has* key indistinguishability*, if no PPT adversary* $\mathcal{A}$ *has more than negligible advantage in the experiment* $\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda)$ *described below:*

1. *Adversary* $\mathcal{A}$ *provides a message* m^* *to puncture at.*
2. *The challenger picks a challenge bit* $b \leftarrow\!\!\$\ \{0,1\}$*. If* $b = 0$*, then* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*. If* $b = 1$*, then* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{PKeyGen}(1^\lambda, m^*)$*. The verification key* vk *is returned to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *gets oracle-access to* $\mathsf{Sign}_{m^*}(\mathsf{sk}, \cdot)$*. The oracle signs any message under the secret key* sk*, except* m^**.*
4. $\mathcal{A}$ *returns a guess bit* b' *and wins iff* $b' = b$*.*

We define the advantage in this experiment by

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{IND\text{-}SPS}} := \left|\Pr[\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda) = 1] - \frac{1}{2}\right|.$$

5.3.2 Compact Signature Witness Encryption

In this section we introduce the notion of a compact t-out-of-n SWE scheme for strongly puncturable signatures.

Compared to the definition of SWE given in Def. 67 in the previous chapter, we have relaxed the definitions here in the following ways:

- The security is non-adaptive as discussed in § 5.2.
- The underlying signature in this work needs to be puncturable and we take multiple signatures instead of one multi-signature as arguments.
- For simplicity, we also focus on the case of encrypting one message with one reference message. An extension is possible by repeatedly executing the protocol, but there is no native support for more efficient batching. This is less critical here, as messages may be of arbitrary polynomial length in our construction.

Definition 78 (Compact Signature-Based Witness Encryption)**.** *A* t*-out-of-*n *cSWE for a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a tuple of two algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *where:*

- $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n), T, m)$*: Encryption takes as input a security parameter* λ*, a set* V *of* n *verification keys of the underlying scheme* Sig*, a reference signing message* T *and a message* m *of arbitrary length* $\ell \in \mathsf{poly}(\lambda)$*. It outputs a ciphertext* ct*.*
- $m \leftarrow \mathsf{Dec}(\mathsf{ct}, (\sigma_i)_{i\in I}, I, V)$*: Decryption takes as input a ciphertext* ct*, a list of signatures* $(\sigma_i)_{i\in I}$*, an index set* $I \subseteq [n]$ *and a set* V *of verification keys of the underlying scheme* Sig*. It outputs a message* m*.*

We require such a scheme to fulfill three properties: correctness, compactness and security.

Definition 79 (Correctness)**.** *A* t*-out-of-*n *cSWE scheme* $\mathsf{cSWE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* correct *if* $\forall\lambda \in \mathbb{N}$*, sets of keys* $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$*, index sets* $I \subseteq [n]$ *with* $|I| \geq t$*, messages* m, T *and signatures* $(\sigma_i)_{i\in I}$ *with* $\mathsf{Sig.Vrfy}(\mathsf{vk}_i, \sigma_i, T) = 1$ *for all* $i \in I$*, it holds* $\mathsf{Dec}(\mathsf{Enc}(1^\lambda, V, T, m), (\sigma_i)_{i\in I}, I, V) = m$*.*

Definition 80 (Compactness)**.** *We consider a cSWE scheme* $\mathsf{cSWE} = (\mathsf{Enc}, \mathsf{Dec})$ *to be compact, if for any* $\lambda \in \mathbb{N}$*, any messages* m, T *and any set of keys* V *of size* $|V| = n$*, it holds that, when we compute a ciphertext* $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, V, T, m)$*, its size* $|\mathsf{ct}|$ *is* $\mathcal{O}(\mathsf{poly}(\lambda, \log n))$*.*

Definition 81 (Selective Security)**.** *A* t*-out-of-*n *cSWE scheme* $\mathsf{cSWE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* (selectively) secure *if for all* $\lambda \in \mathbb{N}$*, all* $t, n = \mathsf{poly}(\lambda), t < n$*, there is no PPT adversary* $\mathcal{A}$ *that has more than negligible advantage in the experiment* $\mathsf{Exp}_{\mathsf{Sec\text{-}Sel}}(\mathcal{A}, 1^\lambda)$ *defined below:*

1. *The adversary* $\mathcal{A}$ *specifies signing reference message* T^* *and indices* $J \subset [n]$ *with* $|J| \leq t - 1$*.*
2. *The experiment generates* n *key pairs for* $i \in [n]$ *as* $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ *and provides* $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ *to* $\mathcal{A}$*, as well as all* sk_i *for* $i \in J$*.*
3. $\mathcal{A}$ *gets to make signing queries for pairs* (i, T)*. If* $i \in J$ *or* $T = T^*$*, the experiment aborts, else it returns* $\mathsf{Sig.Sign}(\mathsf{sk}_i, T)$*.*
4. $\mathcal{A}$ *announces challenge messages* m_0, m_1 *with* $|m_0| = |m_1|$*.*
5. *The experiment flips a bit* $b \leftarrow\!\!\$\ \{0, 1\}$*, and sends* $\mathsf{Enc}(1^\lambda, V, T^*, m_b)$ *to* $\mathcal{A}$*.*
6. $\mathcal{A}$ *gets to make further signing queries for pairs* (i, T)*. If* $i \in J$ *or* $T = T^*$*, the experiment aborts, else it returns* $\mathsf{Sig.Sign}(\mathsf{sk}_i, T)$*.*
7. *Finally,* $\mathcal{A}$ *outputs a guess* b'*.*
8. *If* $b = b'$*, the experiment outputs 1, else 0.*

We define $\mathcal{A}$*'s advantage by* $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Sec\text{-}Sel}} := \left|\Pr\left[\mathsf{Exp}_{\mathsf{Sec\text{-}Sel}}(\mathcal{A}, 1^\lambda) = 1\right] - \frac{1}{2}\right|$*.*

5.4 Compact Threshold SWE for Strongly Puncturable Signatures from $i\mathcal{O}$

In this section, we present a construction of compact SWE for strongly puncturable signatures as defined in Def. 78, which is based on $i\mathcal{O}$ for Turing machines.

5.4.1 Construction

The following scheme works for $n = \mathsf{poly}(\lambda)$ potential signers and requires t-of-n signatures to decrypt.

Given a strongly puncturable signature scheme $\mathsf{Sig}_{\mathsf{punc}}$, our protocol will work for a slightly altered signature scheme $\mathsf{Sig}'_{\mathsf{punc}}$. We define $\mathsf{Sig}'_{\mathsf{punc}}$ to behave exactly as $\mathsf{Sig}_{\mathsf{punc}}$, but additionally the public keys vk output by $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig}'_{\mathsf{punc}}.\mathsf{KeyGen}(1^\lambda)$ have a random part $R_i \leftarrow\!\!\$\ \mathbb{Z}_p$ appended to vk_i, hence its keys are of the form (vk_i, R_i). Let $\ell_{vk} = |(\mathsf{vk}_i, R_i)|$ be the size of its keys, $\mathcal{M}$ its message space, and ℓ_σ be the length of its signatures.

Construction 9. *Towards our construction of a compact t-of-n SWE scheme for* $\mathsf{Sig}'_{\mathsf{punc}}$*, let:*

- $p > 2^\lambda$ *be a prime number with* $\log p = \mathcal{O}(\mathsf{poly}(\lambda))$.
- $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\mu \to \mathbb{Z}_p$ *be a PRF with* $\log n \leq \mu \leq \log p$.
- $\mathsf{PPRF} = (\mathsf{KeyGen}, \mathsf{Punc}, \mathsf{Eval})$ *with* $\mathsf{Eval} : \{0,1\}^\lambda \times \{0,1\}^\mu \to \mathbb{Z}_p$ *be a PPRF and* ℓ_{pkey} *be the size of any key punctured at at most one point.*
- SKE *be a symmetric key encryption scheme*
- SSB *be an SSB-hashing scheme and* $\ell_{\mathsf{hk}}, \ell_\tau, \ell_h$ *be the length of the hashing keys* hk*, proofs* τ *and hashes* h *of* SSB *on input a database* D *with length* n *and with a maximum size of its entries of* ℓ_{vk}.
- $\mathsf{ob}M$ *be an indistinguishability obfuscator for the class of machines* $\mathcal{TM}_\lambda = \{M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')\}$ *where each such machine takes: indices* $i^*, \mathsf{ind}_2 \in \{0, \ldots, n+1\}$*, values* $K, K_1 \in \{0,1\}^\lambda$*,* $h \in \{0,1\}^{\ell_h}, T \in \mathcal{M}, \mathsf{hk} \in \{0,1\}^{\ell_{hk}}$*, a* PPRF *key* $K_2 \in \{0,1\}^\ell_{pkey}$ *and a value* $U \in \mathbb{Z}_p$ *as hardwired inputs and an index* $\mathsf{ind} \in [n]$*, values* vk, R *of combined length* ℓ_{vk}*, as well as* $\tau \in \{0,1\}^{\ell_\tau}$ *and* $\sigma \in \{0,1\}^{\ell_\sigma}$ *as run-time inputs.*
  $M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')$ *is defined as:*
  - *If* $\mathsf{ind} < 1$ *or* $\mathsf{ind} > n$*, abort and output* $\perp$.
  - *If* $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$*, abort and output* $\perp$.
  - *If* $0 = \mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}, \sigma', T)$*, abort and output* $\perp$.
  - *If* $\mathsf{ind} = \mathsf{ind}_2$*: Output* $R + U \mod p$ *and halt.*
  - *If* $\mathsf{ind} \leq i^*$*: Output* $R + \mathsf{PPRF}(K_2, \mathsf{ind}) \mod p$ *and halt.*
  - *Else if* $\mathsf{ind} > i^*$*:*
    - *Write on the tape variables* $i = 1, R = 0, X = 1, S = K$.
    - *While* $i < t$ *: Set* $R = \mathsf{PRF}(K_1, i)$*,* $X = X \cdot \mathsf{ind} \mod p$*,* $S = S + X \cdot R \mod p$*,* $i = i + 1$.
    - *Output* S.

  Note that all these machines have the same description size. Let ℓ_{inp} *be shorthand for its input size and* $T_{\mathcal{TM}}$ *denote the maximum runtime within this machine class. It is syntactically clear that these machines always halt, so* $T_{\mathcal{TM}}$ *is well-defined.*

We will now present the construction of our t-of-n compact SWE scheme.

$\mathsf{cSWE.Enc}(1^\lambda, V = (\mathsf{vk}_i, R_i)_{i\in[n]}, T, m)$:

- *Generate PRF key* $K_1 \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
- *Generate symmetric key* $K \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
- *Generate PPRF key* $K_2 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$.
- *Generate a hashing key* $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, 1)$.
- *Set* $h = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i\in[n]})$.
- *Choose* $U \leftarrow\!\!\$\ \mathbb{Z}_p$.
- *Let* $M' = M_0[K, K_1, h, T, \mathsf{hk}, K_2, U, 0](\cdot, (\cdot, \cdot), \cdot, \cdot)$.
- *Compute* $\mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}(1^\lambda, M', \ell_{inp}, T_{\mathcal{TM}})$.
- *Compute* $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m)$.

- *Output* $(\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$.

$\mathsf{cSWE.Dec}(\mathsf{ct}, (\sigma_i)_{i\in I}, I, V = (\mathsf{vk}_i, R_i)_{i\in[n]})$:

- *Parse* $\mathsf{ct} = (\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$.
- *If* $|I| < t$, *abort.*
- *For* $i \in I$*:*
  - *Compute* $\tau_i = \mathsf{Open}(\mathsf{hk}, V, i)$.
  - *Run* $c_i = \mathsf{ob}M(i, (\mathsf{vk}_i, R_i), \tau, \sigma_i)$.
  - *Notice* $c_i = K + \Sigma_{j=1}^{t-1}\mathsf{PRF}(K_1, j) \cdot i^j$ *i.e. evaluations* $S_i = f(i)$ *on polynomial* $f = K + \Sigma_{j=1}^{t-1}\mathsf{PRF}(K_1, j) \cdot x^j$.
- *Compute* $K' = \sum_{i\in I} c_i \cdot L_i$ *where* $L_i = \prod_{j\in I; j\neq i} \frac{-j}{i-j}$.
- *Output* $m = \mathsf{SKE.Dec}(K', \mathsf{ct}')$.

We point out that by encrypting a symmetric encryption key, we can get an encryption scheme that allows messages m of arbitrary length without impacting the size of $\mathsf{ob}M$.

5.4.2 Proofs

We will now demonstrate that Construction 9 is a compact SWE scheme as defined in Def. 78, proving its correctness, compactness and selective security.

Theorem 21. *Our construction* cSWE *is correct, given that the underlying primitives* $\mathsf{Sig}'_{\mathsf{punc}}$, SKE, SSB *and* $\mathsf{ob}M$ *are correct.*

Proof. Let $\lambda \in \mathbb{N}$, a set of keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, an index set $I \subseteq [n]$ with $|I| \geq t$, messages m, T and signatures $(\sigma_i)_{i\in I}$ be given such that for all $i \in I$, we have $\mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}_i, \sigma_i, T) = 1$. Let us show $\mathsf{Dec}(\mathsf{Enc}(1^\lambda, V, T, m), (\sigma_i)_{i\in I}, I, V) = m$. $\mathsf{Enc}(1^\lambda, V, T, m)$ yields an output $(\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$. In $\mathsf{Dec}((\mathsf{ob}M, \mathsf{ct}', \mathsf{hk}), (\sigma_i)_{i\in I}, I, V)$, we do not abort before calling $\mathsf{ob}M$, as $|I| \geq t$ by requirement. We compute for $i \in I$: $\tau_i = \mathsf{Open}(\mathsf{hk}, V, i)$, $c_i = \mathsf{ob}M(i, (\mathsf{vk}_i, R_i), \tau, \sigma_i)$. By correctness of $\mathsf{ob}M$, this outputs the value generated by $M_0[K, K_1, h, T, \mathsf{hk}, K_2, U, 0](i, (\mathsf{vk}_i, R_i), \tau, \sigma_i)$, which runs the following code:

- If $i < 1$ or $i > n$, abort and output $\perp$.
- If $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{vk}_i, R_i), \tau)$, abort and output $\perp$.
- If $0 = \mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}_i, \sigma_i, T)$, abort and output $\perp$.
- If $i = 0$: Output $R + U \mod p$ and halt.
- If $i \leq 0$: Output $R + \mathsf{PPRF}(K_2, i) \mod p$ and halt.
- Else if $i > 0$:
  - Write on the tape variables $j = 1, R = 0, X = 1, S = K$
  - While $j < t$:
    - Set $R = \mathsf{PRF}(K_1, j)$, $X = X \cdot i \mod p$, $S = S + X \cdot R \mod p$, $j = j + 1$.
  - Output S

And has hardcoded values $K_1 \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$, $K \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$, $K_2 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$, $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, 1)$, $h = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i\in[n]})$.

Clearly $1 \leq i \leq n$. $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{vk}_i, R_i), \tau) = 1$ holds by correctness of SSB since hk, V in Dec are the same as in the call to Enc. $\mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}_i, \sigma_i, T)$ holds by definition.

This means we get for $i \in I$ $c_i = K + \Sigma_{j=1}^{t-1}\mathsf{PRF}(K_1, j) \cdot i^j$ i.e. evaluations $c_i = f(i)$ of a polynomial $f = K + \Sigma_{j=1}^{t-1}\mathsf{PRF}(K_1, j) \cdot x^j$ with $f(0) = K$.

By computing $K' = \sum_{i \in I} c_i \cdot L_i$ with $L_i = \prod_{j \in I; j \neq i} \frac{-j}{i-j}$ being the Langrange polynomials for supporting points $i \in I$ evaluated at 0, we are guaranteed $K' = f(0) = K$.

So Dec finally outputs $\mathsf{SKE.Dec}(K, \mathsf{ct}')$, but since $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m)$, by the correctness of SKE, we output the original message m. □

Theorem 22. *Our construction* cSWE *is compact.*

Proof. The bottleneck of our ciphertext size is the size of the obfuscated Turing machine, which depends polynomially on its description size and polylogarithmically on its runtime. We notice, that we can describe all inputs (runtime as well as hardwired ones) in $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$. All operations (comparisons, modular arithmetic in $\mathbb{Z}_p$, SSB verification, signature verification and (P)PRF evaluations) can be described and evaluated in size/time $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ and we can describe the whole code including the while-loop in $\mathcal{O}(\mathsf{poly}(\log n \cdot \lambda))$, while the maximum runtime is in $\mathcal{O}(n \cdot \mathsf{poly}(\log n \cdot \lambda))$, leading to the desired size.

In more detail, the output of $\mathsf{cSWE}'.\mathsf{Enc}(1^\lambda, (\mathsf{vk}_i, R_i)_{i \in [n]}, T, m)$ is $(\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$, where $\mathsf{hk} = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i \in [n]}), |\mathsf{hk}| = \ell_{hk} = \mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by the efficiency guarantees on SSB.

Further, $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m)$, so $|\mathsf{ct}'| = \mathcal{O}(\mathsf{poly}(\lambda) \cdot |m|) = \mathcal{O}(\mathsf{poly}(\lambda))$. It remains to show that $\mathsf{ob}M$ is in $\mathcal{O}(\mathsf{poly}(\lambda, \log n))$.

$\mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}(1^\lambda, M', \ell_{inp}, T_{\mathcal{TM}})$ with $M' = M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2]$ (with $i^* = 0, \mathsf{ind}_2 = 0$). By the efficiency requirements on $i\mathcal{O}\mathsf{M}$, $|\mathsf{ob}M|$ is therefor of size $\mathcal{O}(\mathsf{poly}(|M'|, \log T_{\mathcal{TM}}, \ell_{inp}, \lambda))$, where $|M'|$ is the description size, $T_{\mathcal{TM}}$ the maximum runtime and ℓ_{inp} the maximum input size of machines in $\mathcal{TM}_\lambda$. The machines in this class are defined as
$M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')$:

- If $\mathsf{ind} < 1$ or $\mathsf{ind} > n$, abort and output $\perp$.
- If $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$, abort and output $\perp$.
- If $0 = \mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}, \sigma', T)$, abort and output $\perp$.
- If $\mathsf{ind} = \mathsf{ind}_2$: Output $R + U \mod p$ and halt.
- If $\mathsf{ind} \leq i^*$: Output $R + \mathsf{PPRF}(K_2, \mathsf{ind}) \mod p$ and halt.
- Else if $\mathsf{ind} > i^*$:
  - Write on the tape variables $i = 1, R = 0, X = 1, S = K$
  - While $i < t$: Set $R = \mathsf{PRF}(K_1, i)$, $X = X \cdot \mathsf{ind} \mod p$, $S = S + X \cdot R \mod p$, $i = i + 1$.
  - Output S

By construction, the sizes of hardwired inputs are as follows: $|i^*|, |\mathsf{ind}_2| = \log n$, $|K|, |K_1| = \lambda$, $|h| = \ell_h = \mathcal{O}(\mathsf{poly}(\lambda))$ by efficiency of SSB, $|T| = \mathcal{O}(\mathsf{poly}(\lambda))$ as $T \in \mathcal{M}$ is from the message space of $\mathsf{Sig}_{\mathsf{punc}}$ with security parameter λ, $|\mathsf{hk}| = \ell_{hk} = \mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by the efficiency guarantees on SSB, $|K_2| = \ell_{pkey} = \mu \cdot \mathsf{poly}(\lambda)$ by the efficiency guarantees of PPRF and $|U| = \log p$.

The *runtime inputs* are bounded in size as follows: $\mathsf{ind} = \log n$, $|(\mathsf{vk}, R)| = \ell_{\mathsf{vk}} = \lambda + \log p$, $|\tau| = \ell_\tau = \mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by efficiency of SSB and $|\sigma| = \ell_\sigma = \mathsf{poly}(\lambda)$.

We can bound $\mu, \log p = \mathcal{O}(\mathsf{poly}(\lambda))$, as we required $\mu \leq \log p = \mathcal{O}(\mathsf{poly}(\lambda))$. So, we see directly that the input size is bounded as

$$\ell_{inp} = \mathcal{O}(\mathsf{poly}(\lambda \cdot \log n)).$$

Towards analysing the *maximum runtime* $T_{\mathcal{TM}}$: Comparisons with ind, i^* can be executed in time $\mathcal{O}(\mathsf{poly}(\log n))$. By efficiency of SSB, the Turing machine code for $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$ has runtime $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$. Further, the code

for $\mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}, \sigma', T)$ has runtime $\mathcal{O}(\mathsf{poly}(\lambda))$. Modular arithmetic in $\mathbb{Z}_p$ can be executed in time $\mathsf{poly}(\log p) = \mathcal{O}(\mathsf{poly}(\lambda))$. $\mathsf{PPRF}(K_2, \mathsf{ind})$ can be executed in $\mathcal{O}(\mu \cdot \mathsf{poly}(\lambda)) = \mathcal{O}(\mathsf{poly}(\lambda))$ by our efficiency requirements on PPRF and so can $\mathsf{PRF}(K_1, i)$. Since $t \leq n$, the while-loop has runtime in $\mathcal{O}(n \cdot \mathsf{poly}(\log n \cdot \lambda))$. So the maximum runtime is bounded by

$$T_{\mathcal{TM}} = \mathcal{O}(n \cdot \mathsf{poly}(\log n \cdot \lambda)).$$

We analyze the *description size* $|M'|$: Comparisons with ind, i^* can be described in size $\mathcal{O}(\mathsf{poly}(\log n))$. The Turing machine code for $\mathsf{SSB}.\mathsf{Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$ has size $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by efficiency of SSB. The code for $\mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}, \sigma', T)$ has size $\mathcal{O}(\mathsf{poly}(\lambda))$. Modular arithmetic in $\mathbb{Z}_p$ can be described in size $\mathsf{poly}(\log p) = \mathcal{O}(\mathsf{poly}(\lambda))$. $\mathsf{PPRF}(K_2, \mathsf{ind})$ can be described in $\mathcal{O}(\mu \cdot \mathsf{poly}(\lambda)) = \mathcal{O}(\mathsf{poly}(\lambda))$ by our efficiency requirements on PPRF and so can $\mathsf{PRF}(K_1, i)$. Since $\log t \leq \log n$, the while-loop can be described in size $\log n + \mathcal{O}(\mathsf{poly}(\log n \cdot \lambda))$. We conclude that the description size is bounded by

$$|M'| = \mathcal{O}(\mathsf{poly}(\log n \cdot \lambda)).$$

Thus, $|\mathsf{ob}M| = \mathcal{O}(\mathsf{poly}(|M'|, \log T_{\mathcal{TM}}, \ell_{inp}, \lambda)) = \mathcal{O}(\mathsf{poly}(\mathsf{poly}(\log n \cdot \lambda), \log(n \cdot \mathsf{poly}(\log n \cdot \lambda)), \mathsf{poly}(\lambda \cdot \log n), \lambda)) = \mathcal{O}(\mathsf{poly}(\log n, \lambda))$.

So the whole ciphertext is in size $\mathcal{O}(\mathsf{poly}(\log n, \lambda))$. □

Theorem 23. *Our construction of* cSWE *is selectively secure, given that* $\mathsf{Sig}_{\mathsf{punc}}$ *is punctured key indistinguishable and puncturable,* SSB *is index hiding and somewhere perfectly binding,* $\mathsf{ob}M$ *is secure,* SKE *is correct, IND-CPA secure and has pseudorandom ciphertexts,* PRF *is a pseudorandom function and* PPRF *is a puncturable pseudorandom function.*

Proof. Let us give a proof sketch first:

We define a series of indistinguishable hybrids and show, that an adversary $\mathcal{A}$ with non-negligible advantage in the last hybrid could break IND-CPA security of SKE.

Our strategy is to puncture all honest keys at message T^*, then gradually move each party's shares from being computed inside the Turing machine into being embedded in the key part R_i in encrypted form. The Turing machine should in the end only have to decrypt the shares from its input and can forget the key K that it was supposed to share. Then, we can observe, that it never decrypts the honest parties' shares, as there is no accepting signature due to puncturability of our signing scheme. We bind the SSB hash to each honest position i^*, puncture the PPRF at i^* to forget the decryption key, hardwire the output share s_{i^*} into the machine instead and replace the share s_{i^*} with an encryption of garbage inside R_{i^*}. Then, we observe, that by SSB binding and puncturability of the signature scheme, no input exists anymore for which s_{i^*} is output and we can forget the hardwired value again. Lastly, we notice that we now need less than t shares of K to run this experiment and can therefore replace them with shares of an unrelated key K' by t-privacy. What remains is a single encryption of m under K, where K is not known or used in any other part of the output.

We turn to the full proof now: Let $\lambda \in \mathbb{N}$, such that $t = \mathsf{poly}(\lambda)$. Let $\mathcal{A}$ be an adversary for $\mathsf{Exp}_{\mathsf{Sec\text{-}Sel}}(\mathcal{A}, 1^\lambda)$ with more than negligible advantage.

Hybrid $\mathcal{H}_0 = \mathcal{H}_1^0$. This game is identical to $\mathsf{Exp}_{\mathsf{Sec\text{-}Sel}}(\mathcal{A}, 1^\lambda)$. To recap, we get indices $J \subset [n]$, $|J| \leq t-1$ and a reference message T^* from $\mathcal{A}$, then the experiment generates key pairs for $i \in [n]$ as $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig}_{\mathsf{punc}}.\mathsf{KeyGen}(1^\lambda)$, $R_i \leftarrow \mathbb{Z}_p$ and provides $V = ((\mathsf{vk}_1, R_1), \ldots, (\mathsf{vk}_n, R_n))$ to $\mathcal{A}$, as well as all sk_i for $i \in J$. We allow

any signing queries for honest keys $\mathsf{vk}_i, i \in [n] \setminus J$ on any message except T^*, before and after the adversary announces challenge messages m_0, m_1 and the experiment responds to this challenge by choosing $b \leftarrow\!\!\$\ \{0,1\}$, and sending $\mathsf{Enc}(1^\lambda, V, T^*, m_b)$ to $\mathcal{A}$. In the end, $\mathcal{A}$ outputs a guess b' to our choice bit b.

Hybrid $\mathcal{H}_1^{i^*}$ for $i^* \in \{1, \ldots, n\}$. In a first hybrid, we change the keys of all honest parties to be punctured at the challenge reference message T^*:

- If $i^* \in J$, this is identical to $\mathcal{H}_1^{i^*-1}$.
- Else, it is identical to $\mathcal{H}_1^{i^*-1}$, except that the key vk_{i^*} is created as a punctured key as $(\mathsf{vk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{Sig}_{\mathsf{punc}}.\mathsf{PKeyGen}(1^\lambda, T^*)$ for the signing reference messsage T^* specified by $\mathcal{A}$ instead of $(\mathsf{vk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{Sig}_{\mathsf{punc}}.\mathsf{KeyGen}(1^\lambda)$.

We observe that our experiment never needs to compute signatures of T^* for honest keys vk_i with $i \notin J$. So clearly, $\mathcal{H}_1^{i^*}$ and $\mathcal{H}_1^{i^*-1}$ are indistinguishable by the punctured-key indistinguishability of $\mathsf{Sig}'_{\mathsf{punc}}$. In the last hybrid $\mathcal{H}_1^n$, all honest keys will be chosen punctured at message T^*. By puncturability of $\mathsf{Sig}_{\mathsf{punc}}$, that means

$$\Pr\left[\exists i \in [n] \setminus J, \exists \sigma \text{ s.t. } \mathsf{Vrfy}(\mathsf{vk}_i^*, \sigma, T^*) = 1\right] = 0.$$

Hybrid $\mathcal{H}_2$. This is identical to $\mathcal{H}_1^n$ except for conceptual changes:

- The reduction computes already at the start of the experiment the keys K, K_1, K_2 it will use in the call to $\mathsf{Enc}(1^\lambda, V, T^*, m_b)$.
- We define the polynomial $f' = K + \Sigma_{j=1}^{t-1} \mathsf{PRF}(K_1, j) \cdot x^j$ and $s_i = f'(i)$ for $i \in [n]$. Note that s_{ind} corresponds to the output of M' on an accepting input $(\mathsf{ind}, (\mathsf{vk}_{\mathsf{ind}}, R_{\mathsf{ind}}), \tau, \sigma')$, where $1 = \mathsf{SSB}.\mathsf{Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}_{\mathsf{ind}}, R_{\mathsf{ind}}), \tau)$ and $1 = \mathsf{Sig}_{\mathsf{punc}}.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{ind}}, \sigma', T)$ for $\mathsf{ind} \in [n]$. The shares s_i for $i \in J$ correspond to the outputs the adversary is guaranteed to get by just signing T^* himself and following the decryption procedure.

We will now loop through $\mathcal{H}_3^i, \mathcal{H}_4^i, \ldots, \mathcal{H}_7^i$ to gradually switch to a setting where the obfuscated machine pulls all of its outputs out of the R_i key parts instead of computing them itself. This is achieved relying on the binding property of SSB to ensure that the adversary can essentially only query the circuit on the correct keys (vk_i, R_i) that it was provided. First, we encode share s_i into R_i, then we use standard tricks of using $i\mathcal{O}$ with punctured PPRF to carefully reprogram the circuit to read s_i from R_i while maintaining input-output behaviour. This is done one share at a time in order to prevent us from having to puncture out multiple indices at the same time, which would lead to an increased circuit size. We start with $\mathcal{H}_7^0 = \mathcal{H}_2$ for notational convenience.

Hybrid $\mathcal{H}_3^{i^*}$ for $i^* \in \{1, \ldots, n\}$. This is identical to $\mathcal{H}_7^{i^*-1}$ (with $\mathcal{H}_7^0 = \mathcal{H}_2$) except that we generate $\mathsf{hk} \leftarrow \mathsf{SSB}.\mathsf{KeyGen}(1^\lambda, n, i^*)$ binding to i^*.

This is clearly indistinguishable from $\mathcal{H}_7^{i^*-1}$ by SSB being index hiding.

We introduce events $\mathbf{bad}_{i^*}^1$ for $i^* \in \{1, \ldots, n\}$. The event $\mathbf{bad}_{i^*}^1$ happens, if the fresh honestly generated key $\mathsf{hk} \leftarrow \mathsf{SSB}.\mathsf{KeyGen}(1^\lambda, n, i^*)$ in $\mathcal{H}_3^{i^*}$ is *not* statistically binding. By SSB being statistically binding and the events being independent the probability of the event $\mathbf{bad}^1$ that at least one of $\mathbf{bad}_{i^*}^1$ occurs can be bounded by

$$\Pr\left[\mathbf{bad}^1\right] = \Pr\left[\exists i^* \in \{1, \ldots, n\} : \mathbf{bad}_{i^*}^1\right] \leq \Sigma_{i^* \in \{1,\ldots,n\}} \Pr\left[\mathbf{bad}_{i^*}^1\right] \leq n \cdot \mathsf{negl},$$

which is negligible. We will condition on $\mathbf{bad}^1$ not happening in the following.

Hybrid $\mathcal{H}_4^{i^*}$ for $i^* \in \{1,\dots,n\}$. This is identical to $\mathcal{H}_3^{i^*}$ except in the way we choose R_{i^*} in vk_{i^*}. We sample $u_{i^*} \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $R_{i^*} = s_{i^*} - u_{i^*} \mod p$ with s_{i^*} defined as above instead of $R_{i^*} \leftarrow \mathbb{Z}_p$.

This is indistinguishable from $\mathcal{H}_3^{i^*}$ as $s_{i^*} - u_{i^*}$ is still uniform in $\mathbb{Z}_p$.

Hybrid $\mathcal{H}_5^{i^*}$ for $i^* \in \{1,\dots,n\}$. This is identical to $\mathcal{H}_4^{i^*}$ except we generate the key $K_{\{i^*\}} \leftarrow \mathsf{PPRF.Punc}(K_2, \{i^*\})$ and the machine M' we use is replaced by $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$ instead of $M' = M_{i^*-1}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$.

Claim 28. *For all $i^* \in \{1,\dots,n\}$ hybrid $\mathcal{H}_5^{i^*}$ is indistinguishable from $\mathcal{H}_4^{i^*}$ by security of $\mathsf{ob}M$.*

Proof. To show this, let us argue, that $M_0 \coloneqq M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$ and $M_1 \coloneqq M_{i^*-1}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$ are functionally equivalent.
We recall that $M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')$ is defined as follows:

- If $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$, abort and output $\perp$.
- If $0 = \mathsf{Sig_{punc}.Vrfy}(\mathsf{vk}, \sigma', T)$, abort and output $\perp$.
- If $\mathsf{ind} = \mathsf{ind}_2$: Output $R + U \mod p$ and halt.
- If $\mathsf{ind} \leq i^*$: Output $R + \mathsf{PPRF}(K_2, \mathsf{ind}) \mod p$ and halt.
- Else if $\mathsf{ind} > i^*$:
  - Write on the tape variables $i = 1, R = 0, X = 1, S = K$
  - While $i < t$:
    - Set $R = \mathsf{PRF}(K_1, i)$, $X = X \cdot \mathsf{ind} \mod p$, $S = S + X \cdot R \mod p$, $i = i + 1$.
  - Output S (Note that $S = f'(\mathsf{ind}) = s_{\mathsf{ind}}$.)

Assuming that there is an input $(\mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*, \sigma^*)$ for which M_0, M_1 produce different outputs, then $1 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*)$, $1 = \mathsf{Sig_{punc}.Vrfy}(\mathsf{vk}^*, \sigma^*, T^*)$, $0 < \mathsf{ind} \leq n$ must hold. Otherwise they both output $\perp$.
We distinguish 3 cases:

$\mathsf{ind}^* > i^*$ In this case both machines output s^*_{ind}.

$\mathsf{ind}^* = i^*$ In this case M_1 outputs s^*_{ind}, but M_0 outputs $R^* + u_{i^*} \mod p$.
We note that $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, i^*)$ and $h = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i\in[n]})$ are honestly created by the reduction and $\mathbf{bad}^1$ hasn't happened. Thus, the SSB key hk is somewhere statistically binding at index i^* and we know that $\mathsf{vk}^* = \mathsf{vk}_{i^*}, R^* = R_{i^*}$.
That means that M_0 outputs $R_{i^*} + u_{i^*} = s_{i^*}$, which is the same output as in M_1.

$\mathsf{ind}^* < i^*$ In this case M_0 outputs $R^* + \mathsf{PPRF}(K_{\{i^*\}}, \mathsf{ind}^*) \mod p$, while M_1 outputs $R^* + \mathsf{PPRF}(K_2, \mathsf{ind}^*) \mod p$.
As $\mathsf{ind} \neq i^*$ and $K_{\{i^*\}} \leftarrow \mathsf{PPRF.Punc}(K_2, \{i^*\})$, we can conclude that these outputs are identical by the PPRF being correct.

It follows that there does not exist an input that makes M_0 and M_1 have a different output, thus $\mathcal{H}_5^{i^*}$ and $\mathcal{H}_4^{i^*}$ are indistinguishable. □

Hybrid $\mathcal{H}_6^{i^*}$ for $i^* \in \{1,\dots,n\}$. This is identical to $\mathcal{H}_5^{i^*}$ except that we now choose $R_{i^*} = s_{i^*} - \mathsf{PPRF}(K_2, i^*) \mod p$ instead of $R_{i^*} = s_{i^*} - u_{i^*} \mod p$ and we use the machine $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$ instead of the previous $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$.

$\mathcal{H}_6^{i^*}$ and $\mathcal{H}_5^{i^*}$ are indistinguishable as PPRF is pseudorandom at the punctured point i^*. Given $K_{\{i^*\}}, i^*$ and either a uniform value $\bar{u} \leftarrow \mathbb{Z}_p$ or $\mathsf{PPRF}(K_2, i^*)$, we can obliviously emulate either $\mathcal{H}_5^{i^*}$ or $\mathcal{H}_6^{i^*}$ to build a distinguisher for pseudorandomness at punctured points of PPRF.

Hybrid $\mathcal{H}_7^{i^*}$ for $i^* \in \{1, \ldots, n\}$. This is identical to $\mathcal{H}_6^{i^*}$ except that we now choose the machine $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$ instead of the previous machine $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$.

Claim 29. *For all $i^* \in \{1, \ldots, n\}$ hybrid $\mathcal{H}_6^{i^*}$ is indistinguishable from $\mathcal{H}_7^{i^*}$ by security of $\mathsf{ob}M$.*

Proof. To show this, let us argue, that $M_0 \coloneqq M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$ and $M_1 \coloneqq M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$ are functionally equivalent.

Assuming that there is an input $(\mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*, \sigma^*)$ for which M_0, M_1 produce different outputs, then $1 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*)$, $1 = \mathsf{Sig_{punc}.Vrfy}(\mathsf{vk}^*, \sigma^*, T^*)$, $0 < \mathsf{ind} \leq n$ must hold.

We distinguish 3 cases:

$\mathsf{ind}^* > i^*$ In this case both machines output s^*_{ind}.

$\mathsf{ind}^* = i^*$ In this case both machines output $R^* + \mathsf{PPRF}(K_2, i^*) \mod p$.

$\mathsf{ind}^* < i^*$ In this case M_0 outputs $R^* + \mathsf{PPRF}(K_2, \mathsf{ind}^*) \mod p$, while M_1 outputs $R^* + \mathsf{PPRF}(K_{\{i^*\}}, \mathsf{ind}^*) \mod p$.
As $\mathsf{ind} \neq i^*$ and $K_{\{i^*\}} \leftarrow \mathsf{PPRF.Punc}(K_2, \{i^*\})$, we can conclude that these outputs are identical by the PPRF preserving functionality under puncturing.

This means such an input resulting in different outputs can not exist; $\mathcal{H}_6^{i^*}$ and $\mathcal{H}_7^{i^*}$ are indeed indistinguishable. □

Hybrid $\mathcal{H}_8$. This is identical to $\mathcal{H}_7^n$, except that we use $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_2, U, 0]$ which is the same as the previous machine, except of it having dummy inputs instead of K, K_1.

As the output of M' does no longer depend on K, K_1 in $\mathcal{H}_7^n$, this is indistinguishable from the previous hybrid by security of $\mathsf{ob}M$.

Hybrid $\mathcal{H}_9$. This is identical to $\mathcal{H}_8$ except that we set the $s_i^* = (K + \Sigma_{j=1}^{t-1} r_j \cdot (i)^j))$ in $R_i = s_i^* - \mathsf{PPRF}(K_2, i) \mod p$ for $i \in [n]$, where we pick $r_j \leftarrow \{0,1\}^\mu$ randomly instead of $s_i^* = (K + \Sigma_{j=1}^{t-1} \mathsf{PRF}(K_1, j) \cdot (i)^j))$.

This is indistinguishable from $\mathcal{H}_8$ by pseudorandomness of PRF.

Now we will almost "revert" the moves in $\mathcal{H}_3$ to $\mathcal{H}_7$ again to put random values into R_i for $i \notin J$, but this time we do not let M' keep the information to decrypt them – we rely on the puncturability of $\mathsf{Sig_{punc}}$ to make sure that the case of decryptions would never have been reached anyways and invoke security of $\mathsf{ob}M$. This deletes all traces of honest shares in the R_i.

Hybrid $\mathcal{H}_{10}^{i^*}$ for $i^* \in \{1, \ldots, n\}$. This is identical to $\mathcal{H}_{13}^{i^*-1}$ (with $\mathcal{H}_{13}^0 = \mathcal{H}_9$) except that we generate $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, i^*)$ binding to i^*.

This is clearly indistinguishable from $\mathcal{H}_{13}^{i^*-1}$ by SSB being index hiding.

We introduce events $\mathbf{bad}^2_{i^*}$ for $i^* \in \{1,\dots,n\}$. The event $\mathbf{bad}^2_{i^*}$ happens, if the fresh honestly generated key $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, i^*)$ in $\mathcal{H}^{i^*}_{10}$ is *not* statistically binding. By SSB being statistically binding and the events being independent the probability of the event $\mathbf{bad}^2$ that at least one of $\mathbf{bad}^2_{i^*}$ occurs can be bounded by

$$\Pr\left[\mathbf{bad}^2\right] = \Pr\left[\exists i^* \in \{1,\dots,n\} : \mathbf{bad}^2_{i^*}\right] \leq \Sigma_{i^*\in\{1,\dots,n\}} \Pr\left[\mathbf{bad}^2_{i^*}\right] \leq n \cdot \mathsf{negl},$$

which is negligible. We will condition on $\mathbf{bad}^2$ not happening in the following.

Hybrid $\mathcal{H}^{i^*}_{11}$ for $i^* \in \{1,\dots,n\}$. If $i^* \in J$, this is identical to $\mathcal{H}^{i^*}_{10}$.
Else, this is identical to $\mathcal{H}^{i^*}_{10}$ except that the machine M' we use is replaced by $M' = M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_{\{i^*\}},\mathsf{PPRF}(K_2,i^*),i^*]$ instead of $M' = M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_2,U,0]$.

This is indistinguishable from $\mathcal{H}^{i^*}_{10}$ by security of $\mathsf{ob}M$ and the argument is analogous to $\mathcal{H}^{i^*}_7$ and $\mathcal{H}^{i^*}_6$ being indistinguishable.

Hybrid $\mathcal{H}^{i^*}_{12}$ for $i^* \in \{1,\dots,n\}$. If $i^* \in J$, this is identical to $\mathcal{H}^{i^*}_{11}$.
Else this is identical to $\mathcal{H}^{i^*}_{11}$ except that we now choose $R_{i^*} = s_{i^*} - u_{i^*} \mod p$ instead of $R_{i^*} = s_{i^*} - \mathsf{PPRF}(K_2,i^*) \mod p$ and $M' = M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_{\{i^*\}},u_{i^*},i^*]$ instead of $M' = M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_{\{i^*\}},\mathsf{PPRF}(K_2,i^*),i^*]$.

$\mathcal{H}^{i^*}_{12}$ and $\mathcal{H}^{i^*}_{11}$ are indistinguishable by PPRF being pseudorandom at punctured points and the argument is analogous to $\mathcal{H}^{i^*}_6$ and $\mathcal{H}^{i^*}_5$ being indistinguishable.

Hybrid $\mathcal{H}^{i^*}_{13}$ for $i^* \in \{1,\dots,n\}$. If $i^* \in J$, this is identical to $\mathcal{H}^{i^*}_{12}$.
This is identical to $\mathcal{H}^{i^*}_{12}$ except that we now choose $M' = M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_2,U,0]$ instead of $M' = M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_{\{i^*\}},u_{i^*},i^*]$.

Claim 30. *For all $i^* \in \{1,\dots,n\}$ hybrid $\mathcal{H}^{i^*}_{13}$ is indistinguishable from $\mathcal{H}^{i^*}_{12}$ by security of $\mathsf{ob}M$.*

Proof. To show this, let us argue, that $M_0 := M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_2,U,0]$ and $M_1 := M_n[\emptyset,\emptyset,h,T^*,\mathsf{hk},K_{\{i^*\}},u_{i^*},i^*]$ are functionally equivalent.

Assuming that there is an input $(\mathsf{ind}^*,(\mathsf{vk}^*,R^*),\tau^*,\sigma^*)$ for which M_0, M_1 produce different outputs, then $1 = \mathsf{SSB.Vrfy}(\mathsf{hk},h,\mathsf{ind}^*,(\mathsf{vk}^*,R^*),\tau^*)$, $1 = \mathsf{Sig_{punc}.Vrfy}(\mathsf{vk}^*,\sigma^*,T^*)$, $0 < \mathsf{ind} \leq n$ must hold.

We distinguish 2 cases:

$\mathsf{ind}^* \neq i^*$ In this case M_0 outputs $R^* + \mathsf{PPRF}(K_2,\mathsf{ind}^*) \mod p$, while M_1 outputs $R^* + \mathsf{PPRF}(K_{\{i^*\}},\mathsf{ind}^*) \mod p$. We can conclude that these outputs are identical by the PPRF preserving functionality under puncturing.

$\mathsf{ind}^* = i^*$ We claim, that no input $(i^*,(\mathsf{vk}^*,R^*),\tau^*,\sigma^*)$ exists where $1 = \mathsf{SSB.Vrfy}(\mathsf{hk},h,i^*,(\mathsf{vk}^*,R^*),\tau^*)$ and $1 = \mathsf{Sig_{punc}.Vrfy}(\mathsf{vk}^*,\sigma^*,T^*)$ hold, hence the output is always $\perp$ in both machines.
Let us assume towards contradiction, that such an input exists. Given that $\mathbf{bad}^2$ didn't happen, the key hk is somewhere statistically binding and we know $\mathsf{vk}^* = \mathsf{vk}_{i^*}$, $R^* = R_{i^*}$. So, it must hold $1 = \mathsf{Sig_{punc}.Vrfy}(\mathsf{vk}_{i^*},\sigma^*,T^*)$ for $i^* \in J$. By puncturability of $\mathsf{Sig_{punc}}$, no such σ^* can exist.

This means such an input with differing output can not exist; $\mathcal{H}^{i^*}_{12}$ and $\mathcal{H}^{i^*}_{13}$ are indeed indistinguishable. □

Hybrid $\mathcal{H}_{14}$. This is identical to $\mathcal{H}_{13}^n$ except that we now compute $R_i \leftarrow\!\!\$\ \mathbb{Z}_p$ for all $i \in [n] \setminus J$ instead of $R_i = s_i - u_i \mod p$ for $u_i \leftarrow\!\!\$\ \mathbb{Z}_p$ being a fresh uniform value for each $i \in [n] \setminus J$.

As the u_{i^*} are not re-used anywhere in the experiment, this is indistinguishable from $\mathcal{H}_{13}^n$ as both distributions lead to uniform R_i for all honest indices $i \in [n] \setminus J$.

It is clear now, that we only ever compute $|J| \leq t-1$ shares of the key K, which are of the from $R_i = K + \Sigma_{j=1}^{t-1} r_j \cdot (i)^j + \mathsf{PPRF}(K_2, i) \mod p$. This clearly corresponds to $|J|$ Shamir shares corresponding to interpolation points $i \in J$ out of a t-of-n sharing for points $i \in [n]$. So in the next hybrid, we can make the key K disappear.

Hybrid $\mathcal{H}_{15}$. This is identical to $\mathcal{H}_{14}$, except that we choose a random $K' \leftarrow \{0,1\}^\lambda$ and make $R_i = K' + \Sigma_{j=1}^{t-1} r_j \cdot (i)^j + \mathsf{PPRF}(K_2, i) \mod p$ for $i \in J$. Where $r_j \leftarrow \mathbb{Z}_p$ randomly.

This is indistinguishable from $\mathcal{H}_{14}$ by the t-privacy of the Shamir secret sharing scheme.

Let us assume now that there is an adversary $\mathcal{A}$ which breaks security of cSWE with probability ε. Conditioned on $\mathbf{bad}^1, \mathbf{bad}^2$ not happening, $\mathcal{A}$ has negligible advantage $n'(\lambda)$ of distinguishing the real experiment from $\mathcal{H}_{15}$.

In hybrid $\mathcal{H}_{15}$, there is no information about K contained anywhere except in the ciphertext $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m_b)$. So we can make an IND-CPA distinguisher $\mathcal{D}$, that simulates $\mathcal{H}_{15}$ by asking the IND-CPA security game of SKE for an encryption with challenge messages m_0, m_1, receiving a ciphertext ct^* and then making all the outputs in $\mathcal{H}_{15}$ honestly, except setting ciphertext part $\mathsf{ct}' = \mathsf{ct}^*$ and outputting whatever $\mathcal{A}$ does.

The advantage we get from this distinguisher in the IND-CPA game is guaranteed to be $\mathsf{Adv}(\mathcal{D}) \geq \varepsilon - n'(\lambda) - \Pr\left[\mathbf{bad}^1\right] - \Pr\left[\mathbf{bad}^2\right] = \varepsilon - \mathsf{negl}(\lambda)$. This means that ε must be negligible due to SKE being IND-CPA secure. □

Bibliography

[ADMM14] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. "Secure Multiparty Computations on Bitcoin". In: *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2014, pp. 443–458. DOI: 10.1109/SP.2014.35.

[ADMSW24] Gennaro Avitabile, Nico Döttling, Bernardo Magri, Christos Sakkas, and Stella Wohnig. "Signature-Based Witness Encryption with Compact Ciphertext". In: *ASIACRYPT 2024, Part I*. Ed. by Kai-Min Chung and Yu Sasaki. Vol. 15484. LNCS. Springer, Singapore, Dec. 2024, pp. 3–31. DOI: 10.1007/978-981-96-0875-1_1.

[AJ15] Prabhanjan Ananth and Abhishek Jain. "Indistinguishability Obfuscation from Compact Functional Encryption". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Berlin, Heidelberg, Aug. 2015, pp. 308–326. DOI: 10.1007/978-3-662-47989-6_15.

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. "1-out-of-n Signatures from a Variety of Keys". In: *ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. LNCS. Springer, Berlin, Heidelberg, Dec. 2002, pp. 415–432. DOI: 10.1007/3-540-36178-2_26.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. "Optimistic Fair Exchange of Digital Signatures (Extended Abstract)". In: *EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. LNCS. Springer, Berlin, Heidelberg, 1998, pp. 591–606. DOI: 10.1007/BFb0054156.

[Bar+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Berlin, Heidelberg, Aug. 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable Delay Functions". In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Cham, Aug. 2018, pp. 757–788. DOI: 10.1007/978-3-319-96884-1_25.

[BCGÜW25] Nicholas Brandt, Miguel Cueto Noval, Christoph U. Günther, Akin Ünal, and Stella Wohnig. *Constrained Verifiable Random Functions Without Obfuscation and Friends*. Cryptology ePrint Archive, Paper 2025/1045. 2025. URL: https://eprint.iacr.org/2025/1045.

[BDDNO21] Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. "TARDIS: A Foundation of Time-Lock Puzzles in UC". In: *EUROCRYPT 2021, Part III*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12698. LNCS. Springer, Cham, Oct. 2021, pp. 429–459. DOI: 10.1007/978-3-030-77883-5_15.

[BDHKS19] Michael Backes, Nico Döttling, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. "Ring Signatures: Logarithmic-Size, No Setup - from Standard Assumptions". In: *EUROCRYPT 2019, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. LNCS. Springer, Cham, May 2019, pp. 281–311. DOI: 10.1007/978-3-030-17659-4_10.

[BDLSY12] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-speed high-security signatures". In: *Journal of Cryptographic Engineering* 2.2 (Sept. 2012), pp. 77–89. DOI: 10.1007/s13389-012-0027-1.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. "Compact Multi-signatures for Smaller Blockchains". In: *ASIACRYPT 2018, Part II*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11273. LNCS. Springer, Cham, Dec. 2018, pp. 435–464. DOI: 10.1007/978-3-030-03329-3_15.

[BDSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. "Noninteractive Zero-Knowledge". In: *SIAM Journal on Computing* 20.6 (1991), pp. 1084–1118. DOI: 10.1137/0220068.

[BDW22] Pedro Branco, Nico Döttling, and Stella Wohnig. "Universal Ring Signatures in the Standard Model". In: *ASIACRYPT 2022, Part IV*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. LNCS. Springer, Cham, Dec. 2022, pp. 249–278. DOI: 10.1007/978-3-031-22972-5_9.

[Ben+20] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. "Can a Public Blockchain Keep a Secret?" In: *TCC 2020, Part I*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12550. LNCS. Springer, Cham, Nov. 2020, pp. 260–290. DOI: 10.1007/978-3-030-64375-1_10.

[Ber+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. "The SPHINCS$^+$ Signature Framework". In: *ACM CCS 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 2129–2146. DOI: 10.1145/3319535.3363229.

[BF01] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Berlin, Heidelberg, Aug. 2001, pp. 213–229. DOI: 10.1007/3-540-44647-8_13.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 103–112. DOI: 10.1145/62212.62222.

[BFOQ25] Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. "BEAT-MEV: Epochless Approach to Batched Threshold Encryption for MEV Prevention". In: *USENIX Security 2025*. Seattle, WA: USENIX Association, Aug. 2025. ISBN: 978-1-939133-52-6. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/bormet.

[BG19] Vitalik Buterin and Virgil Griffith. *Casper the Friendly Finality Gadget*. 2019. arXiv: 1710.09437 [cs.CR].

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. "Functional Signatures and Pseudorandom Functions". In: *PKC 2014*. Ed. by Hugo Krawczyk. Vol. 8383. LNCS. Springer, Berlin, Heidelberg, Mar. 2014, pp. 501–519. DOI: 10.1007/978-3-642-54631-0_29.

[BGLPT15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. "Succinct Randomized Encodings and their Applications". In: *47th ACM STOC*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. ACM Press, June 2015, pp. 439–448. DOI: 10.1145/2746539.2746574.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps". In: *EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. Springer, Berlin, Heidelberg, May 2003, pp. 416–432. DOI: 10.1007/3-540-39200-9_26.

[BH15] Mihir Bellare and Viet Tung Hoang. "Adaptive Witness Encryption and Asymmetric Password-Based Cryptography". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Berlin, Heidelberg, 2015, pp. 308–331. DOI: 10.1007/978-3-662-46447-2_14.

[BHLS25] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. "Exponent-VRFs and Their Applications". In: *Advances in Cryptology – EUROCRYPT 2025*. Ed. by Serge Fehr and Pierre-Alain Fouque. Cham: Springer Nature Switzerland, 2025, pp. 195–224.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. "Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles". In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Berlin, Heidelberg, Mar. 2006, pp. 60–79. DOI: 10.1007/11681878_4.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. "NIZK from LPN and Trapdoor Hash via Correlation Intractability for Approximable Relations". In: *CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. LNCS. Springer, Cham, Aug. 2020, pp. 738–767. DOI: 10.1007/978-3-030-56877-1_26.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing". In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Springer, Berlin, Heidelberg, Dec. 2001, pp. 514–532. DOI: 10.1007/3-540-45682-1_30.

[BLT25] Dan Boneh, Evan Laufer, and Ertem Nusret Tas. *Batch Decryption without Epochs and its Application to Encrypted Mempools*. Cryptology ePrint Archive, Paper 2025/1254. 2025. URL: https://eprint.iacr.org/2025/1254.

[Blu81] Manuel Blum. "Coin Flipping by Telephone". In: *CRYPTO'81*. Ed. by Allen Gersho. Vol. ECE Report 82-04. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981, pp. 11–15.

[BN00] Dan Boneh and Moni Naor. "Timed Commitments". In: *CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. LNCS. Springer, Berlin, Heidelberg, Aug. 2000, pp. 236–254. DOI: 10.1007/3-540-44598-6_15.

[BN06] Mihir Bellare and Gregory Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, 2006, pp. 390–399. DOI: 10.1145/1180405.1180453.

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. "Unrestricted Aggregate Signatures". In: *ICALP 2007*. Ed. by Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki. Vol. 4596. LNCS. Springer, Berlin, Heidelberg, July 2007, pp. 411–422. DOI: 10.1007/978-3-540-73420-8_37.

[Bol03] Alexandra Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme". In: *PKC 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Springer, Berlin, Heidelberg, Jan. 2003, pp. 31–46. DOI: 10.1007/3-540-36288-6_3.

[Bor+25] Jan Bormet, Arka Rai Choudhuri, Sebastian Faust, Sanjam Garg, Hussien Othman, Guru-Vamsi Policharla, Ziyan Qu, and Mingyuan Wang. *BEAST-MEV: Batched Threshold Encryption with Silent Setup for MEV prevention*. Cryptology ePrint Archive, Paper 2025/1419. 2025. URL: https://eprint.iacr.org/2025/1419.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. "Derandomization in Cryptography". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 299–315. DOI: 10.1007/978-3-540-45146-4_18.

[Boy07] Xavier Boyen. "Mesh Signatures". In: *EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. LNCS. Springer, Berlin, Heidelberg, May 2007, pp. 210–227. DOI: 10.1007/978-3-540-72540-4_12.

[BP15] Nir Bitansky and Omer Paneth. "ZAPs and Non-Interactive Witness Indistinguishability from Indistinguishability Obfuscation". In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. LNCS. Springer, Berlin, Heidelberg, Mar. 2015, pp. 401–427. DOI: 10.1007/978-3-662-46497-7_16.

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS 93*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[BR96] Mihir Bellare and Phillip Rogaway. "The Exact Security of Digital Signatures: How to Sign with RSA and Rabin". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 399–416. DOI: 10.1007/3-540-68339-9_34.

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. "New Negative Results on Differing-Inputs Obfuscation". In: *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*. Berlin, Heidelberg: Springer-Verlag, 2016, 792–821. ISBN: 9783662498958.

[Bün+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[But] Vitalik Buterin. *HF1 Proposal*. URL: https://notes.ethereum.org/@vbuterin/HF1_proposal.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. "Indistinguishability Obfuscation from Functional Encryption". In: *56th FOCS*. Ed. by Venkatesan Guruswami. IEEE Computer Society Press, Oct. 2015, pp. 171–190. DOI: 10.1109/FOCS.2015.20.

[BW13] Dan Boneh and Brent Waters. "Constrained Pseudorandom Functions and Their Applications". In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Springer, Berlin, Heidelberg, Dec. 2013, pp. 280–300. DOI: 10.1007/978-3-642-42045-0_15.

[CCNPS23] Andrea Cerulli, Aisling Connolly, Gregory Neven, Franz-Stefan Preiss, and Victor Shoup. *vetKeys: How a Blockchain Can Keep Many Secrets*. Cryptology ePrint Archive, Paper 2023/616. https://eprint.iacr.org/2023/616. 2023. URL: https://eprint.iacr.org/2023/616.

[CDKKN22] Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. "Encryption to the Future - A Paradigm for Sending Secret Messages to Future (Anonymous) Committees". In: *ASIACRYPT 2022, Part III*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13793. LNCS. Springer, Cham, Dec. 2022, pp. 151–180. DOI: 10.1007/978-3-031-22969-5_6.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols". In: *CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. LNCS. Springer, Berlin, Heidelberg, Aug. 1994, pp. 174–187. DOI: 10.1007/3-540-48658-5_19.

[CEG88] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. "An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations". In: *EUROCRYPT'87*. Ed. by David Chaum and Wyn L. Price. Vol. 304. LNCS. Springer, Berlin, Heidelberg, Apr. 1988, pp. 127–141. DOI: 10.1007/3-540-39118-5_13.

[CGPP24] Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. "Mempool Privacy via Batched Threshold Encryption: Attacks and Defenses". In: *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 3513–3529. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri.

[CH91] David Chaum and Eugène van Heyst. "Group Signatures". In: *EUROCRYPT'91*. Ed. by Donald W. Davies. Vol. 547. LNCS. Springer, Berlin, Heidelberg, Apr. 1991, pp. 257–265. DOI: 10.1007/3-540-46416-6_22.

[Cha82] David Chaum. "Blind Signatures for Untraceable Payments". In: *CRYPTO'82*. Ed. by David Chaum, Ronald L. Rivest, and Alan T. Sherman. Plenum Press, New York, USA, 1982, pp. 199–203. DOI: 10.1007/978-1-4757-0602-4_18.

[Cha90] David Chaum. "Showing Credentials without Identification Transferring Signatures between Unconditionally Unlinkable Pseudonyms". In: *AUSCRYPT'90*. Ed. by Jennifer Seberry and Josef Pieprzyk. Vol. 453. LNCS. Springer, Berlin, Heidelberg, Jan. 1990, pp. 246–264. DOI: 10.1007/BFb0030366.

[CHKM10] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. "Comparing two pairing-based aggregate signature schemes". In: *DCC* 55.2-3 (2010), pp. 141–167. DOI: 10.1007/s10623-009-9334-7.

[CHKO08] Jung Hee Cheon, Nicholas Hopper, Yongdae Kim, and Ivan Osipkov. "Provably Secure Timed-Release Public Key Encryption". In: *ACM Trans. Inf. Syst. Secur.* 11.2 (May 2008). ISSN: 1094-9224. DOI: 10.1145/1330332.1330336. URL: https://doi.org/10.1145/1330332.1330336.

[Cho+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. "Laconic Oblivious Transfer and Its Applications". In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Cham, Aug. 2017, pp. 33–65. DOI: 10.1007/978-3-319-63715-0_2.

[CKY09] Jan Camenisch, Aggelos Kiayias, and Moti Yung. "On the Portability of Generalized Schnorr Proofs". In: *EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. LNCS. Springer, Berlin, Heidelberg, Apr. 2009, pp. 425–442. DOI: 10.1007/978-3-642-01001-9_25.

[CL01] Jan Camenisch and Anna Lysyanskaya. "An Efficient System for Nontransferable Anonymous Credentials with Optional Anonymity Revocation". In: *EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. Springer, Berlin, Heidelberg, May 2001, pp. 93–118. DOI: 10.1007/3-540-44987-6_7.

[Clo25] Cloudflare, Inc. *Merkle Town*. Certificate Transparency (CT) log explorer by Cloudflare. Accessed: 10.09.2025. 2025. URL: https://ct.cloudflare.com/.

[CLQ05] Julien Cathalo, Benoît Libert, and Jean-Jacques Quisquater. "Efficient and Non-interactive Timed-Release Encryption". In: *ICICS 05*. Ed. by Sihan Qing, Wenbo Mao, Javier López, and Guilin Wang. Vol. 3783. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 291–303. DOI: 10.1007/11602897_25.

[CM11] Sanjit Chatterjee and Alfred Menezes. "On Cryptographic Protocols Employing Asymmetric Pairings – The Role of Ψ Revisited". In: *Discrete Applied Mathematics* 159.13 (2011), pp. 1311–1322. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2011.04.021.

[CPW20] Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. "Witness Maps and Applications". In: *PKC 2020, Part I*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12110. LNCS. Springer, Cham, May 2020, pp. 220–246. DOI: 10.1007/978-3-030-45374-9_8.

[CRV14] Nishanth Chandran, Srinivasan Raghuraman, and Dhinakaran Vinayagamurthy. *Constrained Pseudorandom Functions: Verifiable and Delegatable*. Cryptology ePrint Archive, Report 2014/522. 2014. URL: https://eprint.iacr.org/2014/522.

[Dam88] Ivan Damgård. "Collision Free Hash Functions and Public Key Signature Schemes". In: *EUROCRYPT'87*. Ed. by David Chaum and Wyn L. Price. Vol. 304. LNCS. Springer, Berlin, Heidelberg, Apr. 1988, pp. 203–216. DOI: 10.1007/3-540-39118-5_19.

[Dat20] Pratish Datta. "Constrained pseudorandom functions from functional encryption". In: *Theoretical Computer Science* 809 (2020), pp. 137–170. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2019.12.004. URL: https://www.sciencedirect.com/science/article/pii/S0304397519307662.

[DDM17] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. "Constrained Pseudorandom Functions for Unconstrained Inputs Revisited: Achieving Verifiability and Key Delegation". In: *PKC 2017, Part II*. Ed. by Serge Fehr. Vol. 10175. LNCS. Springer, Berlin, Heidelberg, Mar. 2017, pp. 463–493. DOI: 10.1007/978-3-662-54388-7_16.

[DDMMT20] Dominic Deuber, Nico Döttling, Bernardo Magri, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. "Minting Mechanism for Proof of Stake Blockchains". In: *ACNS 2020, Part I*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi. Vol. 12146. LNCS. Springer, Cham, Oct. 2020, pp. 315–334. DOI: 10.1007/978-3-030-57808-4_16.

[Des88] Yvo Desmedt. "Society and Group Oriented Cryptography: A New Concept". In: *CRYPTO'87*. Ed. by Carl Pomerance. Vol. 293. LNCS. Springer, Berlin, Heidelberg, Aug. 1988, pp. 120–127. DOI: 10.1007/3-540-48184-2_8.

[DF90] Yvo Desmedt and Yair Frankel. "Threshold Cryptosystems". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 307–315. DOI: 10.1007/0-387-34805-0_28.

[DF92] Yvo Desmedt and Yair Frankel. "Shared Generation of Authenticators and Signatures (Extended Abstract)". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 457–469. DOI: 10.1007/3-540-46766-1_37.

[DFL24] Stefan Dziembowski, Sebastian Faust, and Jannik Luhn. *Shutter Network: Private Transactions from Threshold Cryptography*. Cryptology ePrint Archive, Report 2024/1981. 2024. URL: https://eprint.iacr.org/2024/1981.

[DGSTV18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. "Privacy Pass: Bypassing Internet Challenges Anonymously". In: *PoPETs* 2018.3 (July 2018), pp. 164–180. DOI: 10.1515/popets-2018-0026.

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[DHMW22] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. *McFly: Verifiable Encryption to the Future Made Practical*. Cryptology ePrint Archive, Report 2022/433. 2022. URL: https://eprint.iacr.org/2022/433.

[DHMW23] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. "McFly: Verifiable Encryption to the Future Made Practical". In: *FC 2023, Part I*. Ed. by Foteini Baldimtsi and Christian Cachin. Vol. 13950. LNCS. Springer, Cham, May 2023, pp. 252–269. DOI: 10.1007/978-3-031-47754-6_15.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. "Anonymous Identification in Ad Hoc Groups". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Berlin, Heidelberg, May 2004, pp. 609–626. DOI: 10.1007/978-3-540-24676-3_36.

[DMMNT20] Thomas Dinsdale-Young, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. "Afgjort: A Partially Synchronous Finality Layer for Blockchains". In: *SCN 20*. Ed. by Clemente Galdi and Vladimir Kolesnikov. Vol. 12238. LNCS. Springer, Cham, Sept. 2020, pp. 24–44. DOI: 10.1007/978-3-030-57990-6_2.

[Duc+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme". In: *IACR TCHES* 2018.1 (2018), pp. 238–268. ISSN: 2569-2925. DOI: 10.13154/tches.v2018.i1.238-268. URL: https://tches.iacr.org/index.php/TCHES/article/view/839.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. "A Verifiable Random Function with Short Proofs and Keys". In: *PKC 2005*. Ed. by Serge Vaudenay. Vol. 3386. LNCS. Springer, Berlin, Heidelberg, Jan. 2005, pp. 416–431. DOI: 10.1007/978-3-540-30580-4_28.

[EFR21] Andreas Erwig, Sebastian Faust, and Siavash Riahi. *Large-Scale Non-Interactive Threshold Cryptosystems Through Anonymity*. Cryptology ePrint Archive, Report 2021/1290. 2021. URL: https://eprint.iacr.org/2021/1290.

[ElG85] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074.

[eth22] ethereum.org. *Ethereum 2.0 Keys*. 2022. URL: https://kb.beaconcha.in/ethereum-2-keys.

[FHKS23] Sebastian Faust, Carmit Hazay, David Kretzler, and Benjamin Schlosser. "Statement-Oblivious Threshold Witness Encryption". In: *CSF 2023 Computer Security Foundations Symposium*. IEEE Computer Society Press, July 2023, pp. 17–32. DOI: 10.1109/CSF57540.2023.00026.

[Fis05] Marc Fischlin. "Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors". In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Springer, Berlin, Heidelberg, Aug. 2005, pp. 152–168. DOI: 10.1007/11535218_10.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. "On the Non-malleability of the Fiat-Shamir Transform". In: *INDOCRYPT 2012*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. LNCS. Springer, Berlin, Heidelberg, Dec. 2012, pp. 60–79. DOI: 10.1007/978-3-642-34931-7_5.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. "Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract)". In: *31st FOCS*. IEEE Computer Society Press, Oct. 1990, pp. 308–317. DOI: 10.1109/FSCS.1990.89549.

[Fra90] Yair Frankel. "A Practical Protocol for Large Group Oriented Networks". In: *EUROCRYPT'89*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Vol. 434. LNCS. Springer, Berlin, Heidelberg, Apr. 1990, pp. 56–61. DOI: 10.1007/3-540-46885-4_8.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[Fuc14] Georg Fuchsbauer. "Constrained Verifiable Random Functions". In: *SCN 14*. Ed. by Michel Abdalla and Roberto De Prisco. Vol. 8642. LNCS. Springer, Cham, Sept. 2014, pp. 95–114. DOI: 10.1007/978-3-319-10879-7_7.

[Gar+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *54th FOCS*. IEEE Computer Society Press, Oct. 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13.

[Gen+21] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. "YOSO: You Only Speak Once - Secure MPC with Stateless Ephemeral Roles". In: *CRYPTO 2021, Part II*. Ed. by Tal Malkin and Chris Peikert. Vol. 12826. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 64–93. DOI: 10.1007/978-3-030-84245-1_3.

[GG18] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup". In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 1179–1194. DOI: 10.1145/3243734.3243859.

[GGHK22] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. "Stacking Sigmas: A Framework to Compose Σ-Protocols for Disjunctions". In: *EUROCRYPT 2022, Part II*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13276. LNCS. Springer, Cham, 2022, pp. 458–487. DOI: 10.1007/978-3-031-07085-3_16.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. *Fully Secure Attribute Based Encryption from Multilinear Maps*. Cryptology ePrint Archive, Report 2014/622. 2014. URL: https://eprint.iacr.org/2014/622.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions (Extended Abstract)". In: *25th FOCS*. IEEE Computer Society Press, Oct. 1984, pp. 464–479. DOI: 10.1109/SFCS.1984.715949.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. "Witness encryption and its applications". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 467–476. DOI: 10.1145/2488608.2488667.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". In: *EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Berlin, Heidelberg, May 1999, pp. 295–310. DOI: 10.1007/3-540-48910-X_21.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. "On the (In)security of the Fiat-Shamir Paradigm". In: *44th FOCS*. IEEE Computer Society Press, Oct. 2003, pp. 102–115. DOI: 10.1109/SFCS.2003.1238185.

[GKMPS22] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. "Storing and Retrieving Secrets on a Blockchain". In: *PKC 2022, Part I*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13177. LNCS. Springer, Cham, Mar. 2022, pp. 252–282. DOI: 10.1007/978-3-030-97121-2_10.

[GKPVZ13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. "How to Run Turing Machines on Encrypted Data". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Berlin, Heidelberg, Aug. 2013, pp. 536–553. DOI: 10.1007/978-3-642-40084-1_30.

[GKPW24] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. "Threshold Encryption with Silent Setup". In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. LNCS. Springer, Cham, Aug. 2024, pp. 352–386. DOI: 10.1007/978-3-031-68394-7_12.

[GL89] Oded Goldreich and Leonid A. Levin. "A Hard-Core Predicate for all One-Way Functions". In: *21st ACM STOC*. ACM Press, May 1989, pp. 25–32. DOI: 10.1145/73007.73010.

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. "Lower Bounds on Obfuscation from All-or-Nothing Encryption Primitives". In: *CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. LNCS. Springer, Cham, Aug. 2017, pp. 661–695. DOI: 10.1007/978-3-319-63688-7_22.

[GMR23] Nicolas Gailly, Kelsey Melissaris, and Yolan Romailler. *tlock: Practical Timelock Encryption from Threshold BLS*. Cryptology ePrint Archive, Paper 2023/189. 2023. URL: https://eprint.iacr.org/2023/189.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks". In: *SIAM Journal on Computing* 17.2 (Apr. 1988), pp. 281–308.

[GNB19] Brandon Goodell, Sarang Noether, and Arthur Blue. *Concise Linkable Ring Signatures and Forgery Against Adversarial Keys*. Tech. rep. 2019. URL: https://eprint.iacr.org/2019/654.

[GO94] Oded Goldreich and Yair Oren. "Definitions and Properties of Zero-Knowledge Proof Systems". In: *Journal of Cryptology* 7.1 (Dec. 1994), pp. 1–32. DOI: 10.1007/BF00195207.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Non-interactive Zaps and New Techniques for NIZK". In: *CRYPTO 2006*. Ed. by Cynthia Dwork. Vol. 4117. LNCS. Springer, Berlin, Heidelberg, Aug. 2006, pp. 97–111. DOI: 10.1007/11818175_6.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Berlin, Heidelberg, 2006, pp. 339–358. DOI: 10.1007/11761679_21.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for cryptographers". In: *Discrete Applied Mathematics* 156.16 (2008). Applications of Algebra to Cryptography, pp. 3113–3121. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2007.12.010.

[GR07] Shafi Goldwasser and Guy N. Rothblum. "On Best-Possible Obfuscation". In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, Berlin, Heidelberg, Feb. 2007, pp. 194–213. DOI: 10.1007/978-3-540-70936-7_11.

[GS18] Sanjam Garg and Akshayaram Srinivasan. "A Simple Construction of iO for Turing Machines". In: *TCC 2018, Part II*. Ed. by Amos Beimel and Stefan Dziembowski. Vol. 11240. LNCS. Springer, Cham, Nov. 2018, pp. 425–454. DOI: 10.1007/978-3-030-03810-6_16.

[GVW19] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. "Collusion Resistant Broadcast and Trace from Positional Witness Encryption". In: *PKC 2019, Part II*. Ed. by Dongdai Lin and Kazue Sako. Vol. 11443. LNCS. Springer, Cham, Apr. 2019, pp. 3–33. DOI: 10.1007/978-3-030-17259-6_1.

[Hal10] Jonathan I. Hall. *Generalized Reed-Solomon Codes*. https://users.math.msu.edu/users/halljo/classes/CODENOTES/GRS.pdf. Lecture notes in *Notes on Coding Theory*, Michigan State University. 2010.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1364–1396.

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. "Universal Signature Aggregators". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Berlin, Heidelberg, Apr. 2015, pp. 3–34. DOI: 10.1007/978-3-662-46803-6_1.

[HW15] Pavel Hubacek and Daniel Wichs. "On the Communication Complexity of Secure Function Evaluation with Long Output". In: *ITCS 2015*. Ed. by Tim Roughgarden. ACM, Jan. 2015, pp. 163–172. DOI: 10.1145/2688073.2688105.

[IN83] K. Itakura and K. Nakamura. "A Public-Key Cryptosystem Suitable for Digital Multi-Signatures". In: *NEC Research & Development* 71 (1983), pp. 1–8.

[JDS22] Mei Jiang, Dung Hoang Duong, and Willy Susilo. "Puncturable Signature: A Generic Construction and Instantiations". In: *Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Proceedings, Part II*. Copenhagen, Denmark: Springer-Verlag, Sept. 2022, 507–527. DOI: 10.1007/978-3-031-17146-8_25.

[JJ21] Abhishek Jain and Zhengzhong Jin. "Non-interactive Zero Knowledge from Sub-exponential DDH". In: *EUROCRYPT 2021, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 3–32. DOI: 10.1007/978-3-030-77870-5_1.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability obfuscation from well-founded assumptions". In: *53rd ACM STOC*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM Press, June 2021, pp. 60–73. DOI: 10.1145/3406325.3451093.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". In: *International Journal of Information Security* 1.1 (2001), pp. 36–63. DOI: 10.1007/s102070100002. URL: https://doi.org/10.1007/s102070100002.

[Jou00] Antoine Joux. "A One Round Protocol for Tripartite Diffie-Hellman". In: *Proceedings of the 4th International Symposium on Algorithmic Number Theory*. ANTS-IV. Berlin, Heidelberg: Springer-Verlag, 2000, 385–394. ISBN: 3540676953.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. 2nd. Chapman & Hall/CRC, 2014. ISBN: 1466570261.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. "Indistinguishability Obfuscation for Turing Machines with Unbounded Memory". In: *47th ACM STOC*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. ACM Press, June 2015, pp. 419–428. DOI: 10.1145/2746539.2746614.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. "Delegatable pseudorandom functions and applications". In: *ACM CCS 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM Press, Nov. 2013, pp. 669–684. DOI: 10.1145/2508859.2516668.

[Lev85] Leonid A. Levin. "One-Way Functions and Pseudorandom Generators". In: *17th ACM STOC*. ACM Press, May 1985, pp. 363–365. DOI: 10.1145/22145.22185.

[LJKW18] Jia Liu, Tibor Jager, Saqib A. Kakvi, and Bogdan Warinschi. "How to build time-lock encryption". In: *Des. Codes Cryptogr.* 86.11 (2018), pp. 2549–2586. DOI: 10.1007/s10623-018-0461-x.

[LLC15] Bei Liang, Hongda Li, and Jinyong Chang. "Constrained Verifiable Random Functions from Indistinguishability Obfuscation". In: *ProvSec 2015*. Ed. by Man Ho Au and Atsuko Miyaji. Vol. 9451. LNCS. Springer, Cham, Nov. 2015, pp. 43–60. DOI: 10.1007/978-3-319-26059-4_3.

[LPQ18] Benoît Libert, Thomas Peters, and Chen Qian. "Logarithmic-Size Ring Signatures with Tight Security from the DDH Assumption". In: *ESORICS 2018, Part II*. Ed. by Javier López, Jianying Zhou, and Miguel Soriano. Vol. 11099. LNCS. Springer, Cham, Sept. 2018, pp. 288–308. DOI: 10.1007/978-3-319-98989-1_15.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. "Indistinguishability Obfuscation with Non-trivial Efficiency". In: *PKC 2016, Part II*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9615. LNCS. Springer, Berlin, Heidelberg, Mar. 2016, pp. 447–462. DOI: 10.1007/978-3-662-49387-8_17.

[LS19] Alex Lombardi and Luke Schaeffer. *A Note on Key Agreement and Non-Interactive Commitments*. Cryptology ePrint Archive, Report 2019/279. 2019. URL: https://eprint.iacr.org/2019/279.

[LW15] Arjen K. Lenstra and Benjamin Wesolowski. *A random zoo: sloth, unicorn, and trx*. Cryptology ePrint Archive, Report 2015/366. 2015. URL: https://eprint.iacr.org/2015/366.

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. "Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract)". In: *ACISP 04*. Ed. by Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan. Vol. 3108. LNCS. Springer, Berlin, Heidelberg, July 2004, pp. 325–335. DOI: 10.1007/978-3-540-27800-9_28.

[Lys02] Anna Lysyanskaya. "Unique Signatures and Verifiable Random Functions from the DH-DDH Separation". In: *CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Springer, Berlin, Heidelberg, Aug. 2002, pp. 597–612. DOI: 10.1007/3-540-45708-9_38.

[LZW19] Muhua Liu, Ping Zhang, and Qingtao Wu. "A Novel Construction of Constrained Verifiable Random Functions". In: *Secur. Commun. Networks* 2019 (2019), 4187892:1–4187892:15. DOI: 10.1155/2019/4187892.

[Mad+23] Varun Madathil, Sri Aravinda Krishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. "Cryptographic Oracle-based Conditional Payments". In: *NDSS 2023*. The Internet Society, Feb. 2023.

[Mau05] Ueli M. Maurer. "Abstract Models of Computation in Cryptography (Invited Paper)". In: *10th IMA International Conference on Cryptography and Coding*. Ed. by Nigel P. Smart. Vol. 3796. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 1–12. DOI: 10.1007/11586821_1.

[MGZ23] Peyman Momeni, Sergey Gorbunov, and Bohan Zhang. "FairBlock: Preventing Blockchain Front-Running with Minimal Overheads". In: *Security and Privacy in Communication Networks*. Ed. by Fengjun Li, Kaitai Liang, Zhiqiang Lin, and Sokratis K. Katsikas. Cham: Springer Nature Switzerland, 2023, pp. 250–271. ISBN: 978-3-031-25538-0.

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. "Simple Schnorr multi-signatures with applications to Bitcoin". In: *DCC* 87.9 (2019), pp. 2139–2164. DOI: 10.1007/s10623-019-00608-x.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. "Verifiable Random Functions". In: *40th FOCS*. IEEE Computer Society Press, Oct. 1999, pp. 120–130. DOI: 10.1109/SFFCS.1999.814584.

[MS81] R. J. McEliece and D. V. Sarwate. "On sharing secrets and Reed-Solomon codes". In: *Commun. ACM* 24.9 (Sept. 1981), 583–584. ISSN: 0001-0782. DOI: 10.1145/358746.358762. URL: https://doi.org/10.1145/358746.358762.

[Nat] National Institute of Standards and Technology (NIST). *Post-Quantum Cryptography Standardization Project*. https://csrc.nist.gov/projects/post-quantum-cryptography. Accessed: 2025-08-26.

[Nat23] National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 186-5. Feb. 2023. DOI: 10.6028/NIST.FIPS.186-5. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf.

[Nat24a] National Institute of Standards and Technology (NIST). *Module-Lattice-Based Digital Signature Standard*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 204. Aug. 2024. DOI: 10.6028/NIST.FIPS.204. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf.

[Nat24b] National Institute of Standards and Technology (NIST). *Stateless Hash-Based Digital Signature Standard*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 205. Aug. 2024. DOI: 10.6028/NIST.FIPS.205. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf.

[OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. "New Realizations of Somewhere Statistically Binding Hashing and Positional Accumulators". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Berlin, Heidelberg, 2015, pp. 121–145. DOI: 10.1007/978-3-662-48797-6_6.

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Berlin, Heidelberg, May 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16.

[Ped91] Torben P. Pedersen. "A Threshold Cryptosystem without a Trusted Party (Extended Abstract) (Rump Session)". In: *EUROCRYPT'91*. Ed. by Donald W. Davies. Vol. 547. LNCS. Springer, Berlin, Heidelberg, Apr. 1991, pp. 522–526. DOI: 10.1007/3-540-46416-6_47.

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[Pre+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *FALCON*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022.

[PS19a] Sunoo Park and Adam Sealfon. "It Wasn't Me! - Repudiability and Claimability of Ring Signatures". In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Cham, Aug. 2019, pp. 159–190. DOI: 10.1007/978-3-030-26954-8_6.

[PS19b] Chris Peikert and Sina Shiehian. "Noninteractive Zero Knowledge for NP from (Plain) Learning with Errors". In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Cham, Aug. 2019, pp. 89–114. DOI: 10.1007/978-3-030-26948-7_4.

[PS96] David Pointcheval and Jacques Stern. "Security Proofs for Signature Schemes". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 387–398. DOI: 10.1007/3-540-68339-9_33.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. "Laconic Function Evaluation and Applications". In: *59th FOCS*. Ed. by Mikkel Thorup. IEEE Computer Society Press, Oct. 2018, pp. 859–870. DOI: 10.1109/FOCS.2018.00086.

[RS60] I. S. Reed and G. Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. DOI: 10.1137/0108018.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to Leak a Secret". In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Springer, Berlin, Heidelberg, Dec. 2001, pp. 552–565. DOI: 10.1007/3-540-45682-1_32.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. *Time-Lock Puzzles and Timed-Release Crypto*. Tech. rep. USA: Massachusetts Institute of Technology, 1996.

[RY07] Thomas Ristenpart and Scott Yilek. "The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks". In: *EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. LNCS. Springer, Berlin, Heidelberg, May 2007, pp. 228–245. DOI: 10.1007/978-3-540-72540-4_13.

[Sch90] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 239–252. DOI: 10.1007/0-387-34805-0_22.

[Sha71] Daniel Shanks. "Class number, a theory of factorization, and genera". In: *Proc. of Symp. Math. Soc., 1971*. Vol. 20. 1971, pp. 41–440.

[Sha79] Adi Shamir. "How to Share a Secret". In: *Communications of the Association for Computing Machinery* 22.11 (Nov. 1979), pp. 612–613. DOI: 10.1145/359168.359176.

[Sha84] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. LNCS. Springer, Berlin, Heidelberg, Aug. 1984, pp. 47–53. DOI: 10.1007/3-540-39568-7_5.

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[SW07] Hovav Shacham and Brent Waters. "Efficient Ring Signatures Without Random Oracles". In: *PKC 2007*. Ed. by Tatsuaki Okamoto and Xiaoyun Wang. Vol. 4450. LNCS. Springer, Berlin, Heidelberg, Apr. 2007, pp. 166–180. DOI: 10.1007/978-3-540-71677-8_12.

[SW14] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, 2014, pp. 475–484. DOI: 10.1145/2591796.2591825.

[Tra] Justin Traglia. *Altair The Beacon Chain*. GitHub repository. URL: https://github.com/ethereum/consensus-specs/blob/dev/specs/altair/beacon-chain.md.

[Tso13] Raylin Tso. "A new way to generate a ring: Universal ring signature". In: *Computers & Mathematics with Applications* 65.9 (2013). Advanced Information Security, pp. 1350–1359. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2012.01.039. URL: https://www.sciencedirect.com/science/article/pii/S0898122112000491.

[Yao82] Andrew Chi-Chih Yao. "Theory and Applications of Trapdoor Functions (Extended Abstract)". In: *23rd FOCS*. IEEE Computer Society Press, Nov. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.

[Zhe97] Yuliang Zheng. "Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption)". In: *CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, Berlin, Heidelberg, Aug. 1997, pp. 165–179. DOI: 10.1007/BFb0052234.

[ZLX23] Yao Zan, Hongda Li, and Haixia Xu. "Adaptively Secure Constrained Verifiable Random Function". In: *Science of Cyber Security*. Ed. by Moti Yung, Chao Chen, and Weizhi Meng. Cham: Springer Nature Switzerland, 2023, pp. 367–385. ISBN: 978-3-031-45933-7.