UNIVERSITÄT
DES
SAARLANDES

Saarland University
Faculty for Mathematics and Computer Science
Department of Computer Science

Accountable Infrastructure and Its Impact on Internet Security and Privacy

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

von
Milivoj Simeonovski

Saarbrücken,
2018

Tag des Kolloquiums: 17.09.2019

Dekan: Prof. Dr. Sebastian Hack

Prüfungsausschuss:

Vorsitzender:	Prof. Dr. Markus Bläser
Berichterstattende:	Prof. Dr. Dr. h. c. Michael Backes
	Prof. Dr. Christian Rossow
Akademischer Mitarbeiter:	Dr. Robert Künnemann

Zusammenfassung

Die Internet-Infrastrutur stützt sich auf die korrekte Ausführung zugrundeliegender Protokolle, welche mit Fokus auf Funktionalität entwickelt wurden. Sicherheit und Datenschutz wurden nachträglich hinzugefügt, hauptsächlich durch die Anwendung kryptografischer Methoden in verschiedenen Schichten des Protokollstacks. Fehlende Zurechenbarkeit, eine fundamentale Eigenschaft Handlungen mit deren Verantwortlichen in Verbindung zu bringen, verhindert jedoch, Fehlverhalten zu erkennen und zu unterbinden.

Diese Dissertation betrachtet die Zurechenbarkeit im Internet aus verschiedenen Blickwinkeln. Zuerst untersuchen wir die Notwendigkeit für Zurechenbarkeit in anonymisierten Kommunikationsnetzen um es Proxyknoten zu erlauben Fehlverhalten beweisbar auf den eigentlichen Verursacher zurückzuverfolgen. Zweitens entwerfen wir ein Framework, das die skalierbare und automatisierte Umsetzung des Rechts auf Vergessenwerden unterstützt. Unser Framework bietet Benutzern die technische Möglichkeit, ihre Berechtigung für die Entfernung von Suchergebnissen nachzuweisen. Drittens analysieren wir die Internet-Infrastruktur, um mögliche Sicherheitsrisiken und Bedrohungen aufgrund von Abhängigkeiten zwischen den verschiedenen beteiligten Entitäten zu bestimmen. Letztlich evaluieren wir die Umsetzbarkeit von Hop Count Filtering als ein Instrument DRDoS Angriffe abzuschwächen und wir zeigen, dass dieses Instrument diese Art der Angriffe konzeptionell nicht verhindern kann.

Abstract

The Internet infrastructure relies on the correct functioning of the basic underlying protocols, which were designed for functionality. Security and privacy have been added post hoc, mostly by applying cryptographic means to different layers of communication. In the absence of accountability, as a fundamental property, the Internet infrastructure does not have a built-in ability to associate an action with the responsible entity, neither to detect or prevent misbehavior.

In this thesis, we study accountability from a few different perspectives. First, we study the need of having accountability in anonymous communication networks as a mechanism that provides repudiation for the proxy nodes by tracing back selected outbound traffic in a provable manner. Second, we design a framework that provides a foundation to support the enforcement of the right to be forgotten law in a scalable and automated manner. The framework provides a technical mean for the users to prove their eligibility for content removal from the search results. Third, we analyze the Internet infrastructure determining potential security risks and threats imposed by dependencies among the entities on the Internet. Finally, we evaluate the feasibility of using hop count filtering as a mechanism for mitigating Distributed Reflective Denial-of-Service attacks, and conceptually show that it cannot work to prevent these attacks.

Acknowledgments

Writing a PhD thesis is not a solipsistic process of an individual mind, but rather a reflection of a collaboration between myself and the people that motivated, inspired and supported me. Therefore, I would like to express my sincere gratitude and appreciations to the people that make my whole PhD journey possible.

First, I would like to thank my supervisor Prof. Michael Backes for giving me a chance to do my PhD in his group. He supported me along the way with his invaluable advices and enthusiastic encouragements. He is truly a great mentor, motivator, researcher, and I am very honored to be one of his students.

I also like to thank all my collaborators and co-authors, in particular, the senior ones from whom I got support, guidance and wisdom: Aniket Kate, Rizwan Asghar, Ben Stock, Giancarlo Pellegrino, Christian Rossow, Robert Künnemann. In addition, I hat tip Kim Pecina for his patience and guidance with the ProVerif proofs.

An office is a place where we get inspiration, ideas, and spend most of our time. For this reason, I would like to give special credits to my colleagues Praveen, Sebastian, Sven, Olli, Malte, Jie, Marie-Therese, Aurore whom I enjoyed sharing an office over the years. Although Yang was never my office mate, I thank him for stopping by on a regular basis and cheering us up with his unique sense of humor. Additionally, I would like to express my sincere gratitude to all my colleagues and friends at CISPA for creating a joyful working environment. Special thanks to our team assistant Bettina for making our days brighter and keeping the group running smoothly.

To maintain a healthy balance between the mind and the body, the recognition goes to my enthusiastic tennis buddies Kiril, Dragan, Bojan, Sebastian, Kim, Hazem. Thank you guys for letting me win from time to time.

Saarbrücken is the place where many friendships were born or strengthened. I am grateful to my best friends in Saarbrücken for their love and support: Kiril, Bojan, Dragan, Monika, Evica, Marinela. Special thanks to Monika who gave me the idea and encouragement to come to Saarbrücken and introduced me to my wife. I am fortunate to have many other friends supporting me through the years: Irena, Jasmina, Diana, Mohamed, Christina, Monica, Lisette, Edite, Nisa, Varvara, Max, Iulia, Tomas, Saskia, Freddie, Christine, Levi. My sincere gratitude to my best friends in Macedonia: Filip, Bojan, Aleksandra, Neda who always are standing by my side with their unconditional support.

Despite being mentioned at the end, the support of the family is the most important thing that truly drives and motivates. I am deeply grateful to my parents and my brother for their support and encouragement. Without them, I would certainly not stand where I stand today. Mainly, I thank my wife Tanja and our daughter Amelia for their faith in me and unconditional love that I get every single moment.

Contents

List of Figures

List of Tables

List of Listings

1

Background of this Dissertation

Underlying Scientific Papers

The foundation of this dissertation are published, peer-reviewed scientific publications where I contributed as one of the main authors. The following list gives a short summary of the publications along with the clarification of the authorship of the respective papers.

1. Many anonymous communication networks rely on routing traffic through proxy nodes to obfuscate the originator of the traffic. Without an accountability mechanism, exit proxy nodes risk sanctions by law enforcement if users commit illegal actions through the AC network. Chapter 3 presents BackRef, a generic mechanism for AC networks that provides practical repudiation for the proxy nodes by tracing back the selected outbound traffic to the predecessor node (but not in the forward direction) through a cryptographically verifiable chain. This work was published in the proceedings of the 12th International Conference on Applied Cryptology and Network Security (ACNS) in 2015 [1]. Aniket Kate and I developed the main concept of BackRef. Peter Druschel contributed in solving the last mile problem. All authors contributed with general writing tasks and performed reviews of the paper.

2. Indexing systems such as Google search, collect, store, and organize publicly available data in order to provide accurate information retrieval for the end users. This data, typically comprise a variety of sources that contain personal information, often with detrimental effects on the individual's privacy. To grant individuals the ability to regain control over their disseminated personal information, the European Court of Justice ruled that EU citizens have a *right to be forgotten* in the sense that indexing systems, must offer them technical means to request removal of links from search results. Chapter 4 presents Oblivion, a universal framework to support the automation of the right to be forgotten in a scalable, provable and privacy-preserving manner. This work was published in proceedings of the 13th International Conference on Applied Cryptology and Network Security (ACNS) in 2015 [2]. Rizwan Asghar and I mainly developed the concept of Oblivion. The security analysis part was conducted by Fabian Bendun. All authors contributed with general writing tasks and performed reviews of the paper.

3. In the current Internet, there is not much accountability in place which highlights and quantifies the need for accountable behavior among the entities on the Internet. Chapter 5 present an analysis of the Internet infrastructure determining potential threats imposed by dependencies between the entities involved. We analyse dependencies among the entities on the Internet responsible for the Internet's core services as well as the dependences that are indirectly imposed by the service providers that distribute content such as JavaScript. In this work, we present a technique for modeling services,

providers, and dependencies as a property graph, and then reason on global-scale threats. Our what-if analysis quantifies the impact of security incidents when providers/entities are set to be attackers but also victims. This work was published in the proceedings of the 26th International Conference on World Wide Web (WWW) in 2017 [3]. Giancarlo Pellegrino and I are the two main co-authors of the paper that developed and realized the idea. I was responsible for the major part of the implementation and evaluation. All authors performed reviews of the paper.

4. In a scenario of a denial of service attack, in particular, DRDoS attack, service providers play an important role in protecting the Internet infrastructure as well as their users. Led by the idea that an attacker cannot fabricate the number of hops a packet travels between amplifier and victim, Hop Count Filtering mechanisms that analyze the Time-to-Live (TTL) of incoming packets have been proposed as a solution for mitigating DRDoS attacks. In Chapter 6, we present our work on the feasibility of such a mechanism for DRDoS mitigation. We analyze the potential of this mechanism from two different perspectives, namely the defender's and the attacker's perspective. First, we detail how a server can use active probing to learn the correct TTLs of the alleged packet and consequently filter the spoofed packages. Then, from the attacker's perspective, we evaluate the TTL mechanism and show that a spoofing attacker can subvert TTL-based filters by predicting the TTL value. Our findings conceptually show that TTL-based defenses cannot work to thwart the outlined attacks. This work was published in the proceedings of the 19th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID) in 2016 [4]. The general idea for this work was developed in a joint effort among all the authors. Teemu Rytilahtiu mainly contributed by re-evaluating the concept of Hop Count Filtering (with strong support from Ben Stock) to determine the necessary level of tolerance required for the approach to work in practice. I developed the methodology for predicting the TTL values. All authors contributed with general writing tasks and performed reviews of the paper.

The following table (Table 1.1) lists the peer-reviewed scientific publications that I have published, grouped by years, while the highlighted ones are the underlying scientific publications of this thesis.

2018	Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. "A Survey on Routing in Anonymous Communication Protocols." In: *ACM Computing Surveys (CSUR)* 51.3 (2018), 51:1–51:39
	Muhammad Rizwan Asghar, Michael Backes, and Milivoj Simeonovski. "PRIMA: Privacy-Preserving Identity and Access Management at Internet-Scale." In: *Proceedings of the IEEE International Conference on Communications (ICC 2018)*. IEEE, 2018
	Patrick Speicher, Marcel Steinmetz, Robert Künnemann, Milivoj Simeonovski, Giancarlo Pellegrino, Joerg Hoffmann, and Michael Backes. "Formally Reasoning about the Cost and Efficacy of Securing the Email Infrastructure." In: *Proceedings of the 3nd IEEE European Symposium on Security and Privacy (Euro S&P 2018)*. IEEE, 2018
2017	Milivoj Simeonovski, Giancarlo Pellegrino, Christian Rossow, and Michael Backes. "Who Controls the Internet?: Analyzing Global Threats using Property Graph Traversals." In: *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*. ACM
2016	Michael Backes, Thorsten Holz, Christian Rossow, Teemu Rytilahti, Milivoj Simeonovski, and Ben Stock. "On the Feasibility of TTL-Based Filtering for DRDoS Mitigation." In: *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2016)*. Springer, 2016
2015	Milivoj Simeonovski, Fabian Bendun, Muhammad Rizwan Asghar, Michael Backes, Ninja Marnau, and Peter Druschel. "Oblivion: Mitigating Privacy Leaks by Controlling the Discoverability of Online Information." In: *Proceedings of the 13th International Conference on Applied Cryptography and Network Security (ACNS 2015)*. 2015
2014	Milivoj Simeonovski. "POSTER: Quasi-ID: In Fact, I Am a Human." In: *Proceedings of the 21st ACM Conference on Computer and Communication Security (CCS 2014)*. ACM, 2014
	Michael Backes, Jeremy Clark, Aniket Kate, Milivoj Simeonovski, and Peter Druschel. "BackRef: Accountability in Anonymous Communication Networks." In: *Proceedings of the 12th International Conference on Applied Cryptography and Network Security (ACNS 2014)*. 2014

Table 1.1: Published peer-reviewed papers.

2
Introduction

The Internet has been evolving rapidly over time. From a network primarily used for communication to a platform that has become an integral part of everyday life. It has revolutionized businesses and has changed the way how people interact. In addition to the social benefits that this evolution has brought, the extensive reach of the Internet has created new security and privacy threats. Historically, the Internet was designed for functionality, without paying too much attention neither to security nor privacy. Security and privacy have been added post hoc, mostly by applying cryptographic means to various layers of communication.

The Internet infrastructure relies on the correct functioning of the basic underlying protocols, along with the accountable behavior of the entities that are responsible for the core Internet operations such as routing, name resolution, email transfers, *etc.* In the absence of accountability, as a first order property, the Internet infrastructure does not have a built-in ability to detect and/or prevent certain types of malicious behavior. As a result, the overall security of the Internet depends on or at least is influenced by the behavior of the entities that operate on different communication layers. For example, service providers can perform various attacks such as advertising false BGP paths to sensitive targets through their network [9, 10] and injecting HTTP responses into TCP connections [11]. A malicious service provider can deliver malicious content to the end users which then can be used to mount different type of attacks, *e.g.,* The Great Cannon attack [12].

Furthermore, protecting privacy on the Internet is a widely unsolved challenge for the users and providers. On the one hand, users tend to reveal personal information without considering the widespread of online data, on the other hand, service providers collect users' data in order to understand their behavior and improve the services they provide. The current Internet lacks accountability as a foundational property, hence the users have to place reliance on the service providers for their data. Once the data is disclosed or leaked, the widespread is inevitable because of the absence of a fundamental ability to associate an action with the responsible entity. The implications reported in the press range from public embarrassment and loss of prospective opportunities to safety issues. Since the prevention of the data widespread is impossible, the privacy risks can be at least mitigated by controlling the findability of the disclosed information.

In this dissertation, we look at accountability from a few different perspectives. We start with a design of a framework for adding accountability mechanism to anonymous communication networks in order to protect the proxy nodes from false accusation. Then we continue by proposing an accountable infrastructure for the users and indexing systems, where the users can control the findability of their disseminated data. Finally, we analyse the importance of the accountable behavior among the entities on the Internet.
The thesis consists of four different parts in the following order.

BackRef. The nature of the properties of the technology behind the anonymous communication networks can sometimes be harmful for the nodes serving as proxies. Without an accountability mechanism, exit proxy nodes that forward the traffic may become embroiled in a criminal investigation if the originating user commits criminal actions through the anonymous communication network.

We present BackRef [1], a design of an accountable infrastructure for anonymous communication networks. BackRef is a generic mechanism that provides practical repudiation for the proxy nodes by tracing back the selected outbound traffic to the predecessor node (but not in the forward direction) through a cryptographically verifiable chain. It also provides an option for full (or partial) traceability back to the entry node or even to the corresponding originator when all intermediate nodes are cooperating. To maintain a good balance between anonymity and accountability, the protocol incorporates whitelist directories at exit proxy nodes. Moreover, to assist in the design of BackRef, we introduce a novel concept of pseudonymous signatures that may be of independent interest.

Oblivion. Search engines (or indexing systems) are the prevalently used tools to collect information about individuals on the Internet by means of crawling the entire web space. Search results typically comprise a variety of sources that contain personal information — either intentionally released by the person herself, or unintentionally leaked or published by third parties without being noticed, often with detrimental effects on the individual's privacy. To grant individuals the ability to regain control over their disseminated personal information, the European Court of Justice ruled that EU citizens have a right to be forgotten in the sense that indexing systems, such as Google, must offer them technical means to request removal of links from search results that point to sources violating their data protection rights. These technical means consist of a web form that requires a user to manually identify all relevant links herself upfront and to insert them into the web form, followed by a manual evaluation by employees of the indexing system to assess if the request to remove those links is eligible and lawful.

We present Oblivion [2], a universal framework for supporting the automation of the right to be forgotten in a scalable, provable, and privacy-preserving manner. Oblivion enables a user to automatically find and tag her disseminated personal information using natural language processing (NLP) and image recognition techniques and file a request in a privacy-preserving manner. Second, Oblivion provides indexing systems with an automated and provable eligibility mechanism, asserting that the author of a request is indeed affected by an online resource. The automated eligibility proof ensures censorship-resistance so that only legitimately affected individuals can request the removal of the corresponding links from the search results.

Who Controls the Internet? The Internet is built on top of the basic network services, such as, routing, DNS, email, and CDNs which are operated and controlled by private or governmental organizations. These organizations are usually held accountable for the correct functioning of the services they provide because, on one hand, they serve a huge portion of the Internet users, and on the other hand, their business model depends on the proper functioning of the services. However, the more important role an organization has in the Internet's ecosystem, the more likely is to be involved in security-related incidents. Recent events have shown that these organizations may, knowingly or unknowingly, be part of global-scale incidents including large-scale DDoS attacks and state-sponsored mass surveillance programs.

The research community has been constantly studying the security of the Internet infrastructure considering attack techniques and root cause vulnerabilities. However, we lack models and algorithms to study the intricate dependencies between services and providers, reason on their abuse, and assess the attack impact. To close this gap, we present a technique [3] that models services, providers, and dependencies as a property graph. Moreover, we present a taint-style propagation-based technique to query the model and present an evaluation of our framework on the top 100K Alexa domains.

TTL-based Filtering. Distributed Reflective Denial-of-Service attacks are one of the main disruptions for network providers in the recent years. The adversary spoofs the IP address of a victim and sends a flood of tiny packets to vulnerable services. The services then respond to the spoofed IP, flooding the victim with large replies. In a scenario of such an attack, the service providers play an important role in protecting the Internet infrastructure as well as their users.

Following the idea that an attacker cannot fabricate the number of hops a packet travels between amplifier and victim, Hop Count Filtering mechanisms that utilize the Time-to-Live (TTL) of incoming packets have been proposed as a solution. Our work [4] evaluates the feasibility of using Hop Count Filtering for mitigating Distributed Reflective Denial-of-Service attacks. First, we detail how a server can use active probing to learn TTLs of alleged packet senders. Based on data sets of benign and spoofed Network Time Protocol (NTP) requests, we find that a TTL-based defense could block over 75% of spoofed traffic while allowing 86.4% of benign traffic to pass. To achieve this performance, however, such an approach must allow for a tolerance of +/-2 hops. Second, we investigate the tacit assumption that an attacker cannot learn the correct TTL value. By using a combination of tracerouting and BGP data, we build statistical models which allow to estimate the TTL within that tolerance level. We observe that by wisely choosing the used amplifiers, the attacker is able to circumvent such TTL-based defenses.

Outline

The remainder of this dissertation is structured as follows. We present BackRef in Chapter 3 and Oblivion in Chapter 4. Who Controls the Internet? in Chapter 5 and TTL-based Filtering is presented in Chapter 6. We conclude this dissertation in Chapter 7.

3 BackRef

Accountability in Anonymous Communication Networks

3.1 Motivation

Anonymous communication networks (ACNs) are designed to hide the originator of each message within a larger set of users. In some systems, like DC Nets [13] and Dissent [14], the message emerges from aggregating all participants' messages. In other systems, like onion routing [15], mix networks [16, 17, 18, 19], and peer-to-peer anonymous communication networks [20, 21], messages are routed through volunteer nodes that act as privacy-preserving proxies for the users' messages. In this thesis, we focus on the latter class which is also known as proxy-based anonymous communication networks.

Proxy-based anonymous communication networks provide a powerful service to their users, and correspondingly they have been the most successful ACNs so far. However, the nature of the properties of the technology can sometimes be harmful to the nodes serving as proxies. If a network user's online communication results in a criminal investigation or a cause of action, the last entity to forward the traffic may become embroiled in the proceedings [22, 23, 24, 25], whether as the suspect/defendant or as a third party with evidence. Repudiation in the form of a partial or full traceability has never been a component of any widely-deployed ACN, however, it may become the case that new anonymity networks, or a changing political climate, initiate an interest in providing a verifiable trace to users who misuse anonymity networks according to laws or terms of service.

While several proposals [26, 27, 28, 29, 30, 31, 32] have been made to tackle or at least to mitigate this problem under the umbrella term of *accountable anonymity*, as we discuss in the next section some of them are broken, while others are not scalable enough for deploying in low latency ACNs.

3.2 Contributions

In this work, we design BackRef, a novel practical repudiation mechanism for anonymous communication, which has advantages in terms of deployability and efficiency over the literature. To assist in the design of BackRef, we propose a concept of pseudonymous signatures, which employ pseudonyms (or half Diffie-Hellman exponents) as temporary public keys (and corresponding temporary secrets) employed or employable in almost all ACNs for signing messages. These pseudonym signatures are used to create a verifiable *pseudonym-linkability* mechanism where any proxy node within the route or path, *when required*, can verifiably reveal its predecessor in a time-bound manner. We use this property to design a novel repudiation mechanism, which allows each proxy node, in cooperation with the network, to issue a cryptographic guarantee that a selected traffic flow can be traced back to its originator (*i.e.,* predecessor node) while maintaining the eventual forward secrecy of the system.

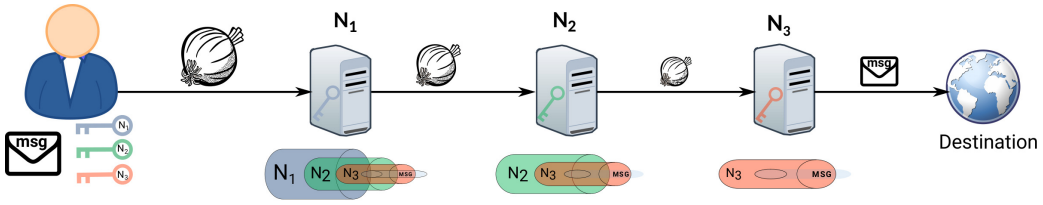


Figure 3.1: The concept of onion routing

Unlike the related work, which largely relies on group signatures and/or anonymous credentials, BACKREF avoids the logistical difficulties of organizing users into groups and arranging a shared group key, and does not require access to a trusted party to issue credentials. While BACKREF is applicable to all proxy-based ACNs, we illustrate its utility by applying it to the onion routing (OR) protocol. We observe that it introduces a small computational overhead and does not affect the performance of the underlying OR protocol. BACKREF also includes a *whitelisting* option; *i.e.,* if an exit node considers traceability to one or more web-services unnecessary, then it can include those services in a *whitelist* directory such that accesses to those are not logged.

We formally define the important properties of the BACKREF network. In particular, we formalize anonymity and no forward traceability as observational equivalence relations, and backward traceability and no false accusation as trace properties. We conduct a formal security analysis of BACKREF using ProVerif, an automated cryptographic protocol verifier, establishing the aforementioned security and privacy properties against a strong adversarial model. We believe both the definitions and the security analysis are of independent interest since they are the first for the OR protocol.

3.3 Background and Related Work

Anonymous communication networks aim at protecting personally identifiable information (PII), in particular, the network addresses of the communicating parties by hiding correlation between input and output messages at one or more network entities. For this purpose, the ACN protocols employ techniques such as using a series of intermediate routers and layered encryptions to obfuscate the source of a communication, and adding fake traffic to make the "real" communication difficult to extract.

3.3.1 Anonymous Communication Protocols

Single-hop proxy servers, which relay traffic flows, enable a simple form of anonymous communication. However, anonymity in this case requires, at a minimum, that the proxy is trustworthy and not compromised. Nevertheless, this approach

does not protect the anonymity of senders if the adversary inspects traffic through the proxy [33]. Even with the use of encryption between the sender and the proxy server, timing attacks can be used to correlate flows.

Starting with Chaum [16], several technologies for anonymous communication have been developed in the last thirty years to provide stronger anonymity not dependent on a single entity [13, 14, 15, 17, 18, 34, 35, 36, 37, 38, 39, 40]. Among these, mix networks [16, 17] and onion routing [34] have arguably been most successful. Both offer user anonymity, relationship anonymity, and unlinkability [41], but they obtain these properties through differing assumptions and techniques.

An onion routing (OR) infrastructure involves a set of *routers* (or *OR nodes*) that relay traffic, a *directory service* providing status information for OR nodes, and *users*. Users benefit from anonymous access by constructing a *circuit*—a small ordered subset of OR nodes—and routing traffic through it sequentially (Figure 3.1). The crucial property for anonymity is that an OR node within the built circuit is not able to identify any portion of the circuit other than its predecessor and successor. The user sends messages (to the first OR node in the circuit) in a form of an *onion*—a data structure multiply encrypted by symmetric session keys (one encryption layer per node in the circuit). The symmetric keys are negotiated during an initial *circuit construction* phase. This is followed by a second phase of *low latency* communication (opening and closing streams) through the constructed circuit for the session duration. An OR network does not aim at providing anonymity and unlinkability against a global passive observer, which in theory can analyze end-to-end traffic flow. Instead, it assumes an adversary that adaptively compromises a small fraction of OR nodes and controls a small fraction of the network.

A mix network (Figure 3.2) achieves anonymity by relaying messages through a path of mix nodes (or mixes) in a latency-tolerant manner. The user encrypts a message to be partially decrypted by each mix along the path. Mixes accept a batch of encrypted messages, which are partially decrypted, randomly permuted, and forwarded. Unlike onion routing, an observer is unable to correlate incoming

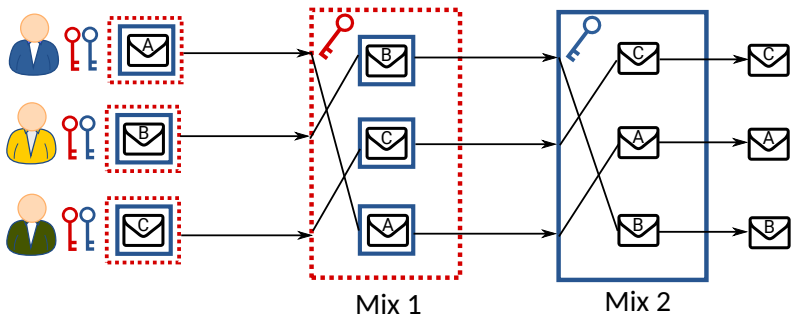


Figure 3.2: The concept of mix network (Mix cascade with two mixes)

and outgoing messages at the mix node; thus, mix networks provide anonymity against a powerful global passive adversary. In fact, as long as a single mix node in the user's path remains uncompromised, the message will maintain some anonymity. However, batching of messages at a mix node introduces inherent delays, making mix networks unsuitable for low-latency, interactive applications (*e.g.,* web browsing, instant messaging). When used, it is for latency-tolerant applications like anonymous email.

3.3.2 Accountable Anonymity Mechanisms

The literature has examined several approaches for adding accountability to ACN technologies, allowing misbehaving users to be selectively traced [26, 27, 28], exit nodes to deny originating traffic it forwards [29, 30], misbehaving users to be banned [31, 32], and misbehaving participants to be discovered [14, 42, 43]. All of these approaches either require users to obtain credentials or do not extend to interactive, low-latency, internet-scale ACNs. A number of them also partition users into subgroups, which reduces anonymity and requires a group manager. BackRef does not require credentials, subgroups, and it is compatible with low-latency ACNs like onion routing, adding minimal overhead.

Kopsell *et al.* [26] propose traceability through threshold group signatures. A user logs into the system to join a group, signs messages with a group signature, and a group manager is empowered to revoke anonymity. The system also introduces an external proxy to inspect all outbound traffic for correct signatures and protocol compliance. The inspector has been criticized for centralizing traffic flows, which enables DoS, censorship, and increases observability [44].

Von Ahn *et al.* [27] also use group signatures as the basis for a general transformation for traceability in ACNs and illustrate it with DC Nets. Users are required to register as members of a group capable of sending messages through the network. Our solution can be viewed as a follow-up to this paper, with a focus on deployability: we do not require users to be organized into groups or introduce new entities, and we concentrate on onion routing.

Diaz and Preneel [28] achieve traceability through issuing anonymous credentials to users and utilizing a traitor tracing scheme to revoke anonymity. It is tailored to high-latency mix networks and requires a trusted authority to issue credentials—both impede deployability. Danezis and Sassaman [44] demonstrate a bypass attack on this and the Kopsell *et al.* scheme [26]. The attack is based on the protocols' assumption that there can be no leakage of information from inside the channel to the world unless it passes through the verification step. Our protocol does not rely on such a strong assumption, namely any exit node (or any node who leaks the information) with enabled BackRef can always activate the repudiation mechanism and shift liability to its predecessor node.

Short of revoking the anonymity of misbehaving users, techniques have been

proposed to at least allow exit nodes to deny originating the traffic. Golle [29] and Clark *et al.* [30] pursue this goal, with the former being specific to high-latency mix networks and the latter requiring anonymous credentials. Tor offers a service called ExoneraTor [45] that provides a record of which nodes were online at a given time, but it does not explicitly prove that a given traffic flow originated from Tor. Other techniques, such as Nymble [31] and its successors (see a survey [32]), enable users to be banned. However these systems inherently require some form of credential or pseudonym infrastructure for the users, and also require web-servers to verify user requests. Finally, Dissent [14] and its successors [42, 43] presents an interesting approach for accountable anonymous communication for DC Nets [13], however even when highly optimized [42], DC Nets are not competitive for internet-scale application.

3.4 Design Overview

In this section we describe our threat model and system goals, and present our key idea and design rationale.

3.4.1 Threat Model and System Goals

We consider the same threat model as the underlying ACN in which we wish to incorporate the BackRef mechanism. Our active adversary $\mathcal{A}$ aims at breaking some anonymity property by determining the ultimate source and/or destination of a communication stream or breaking unlinkability by linking two communication streams of the same user. We assume that some, but not all, of the nodes in the path of the communication stream are compromised by the adversary $\mathcal{A}$, who knows all their secret values, and is able to fully control their functionalities. We, however, limit the attacker, such that he does not have control over both entry and exit node, and cannot correlate end-to-end traffic [46, 47].

For high latency ACNs like mix networks, we assume that the adversary can also observe all traffic in the network, as well as intercept and inject arbitrary messages, while for low latency ACNs like onion routing, we assume the adversary can observe, intercept, and inject traffic in some parts of the network.

While maintaining the anonymity and unlinkability properties of an ACN, we wish to achieve the following goals when incorporating BackRef in the ACN:

Repudiation: For a communication stream flowing through a node, the node operator should be able to prove that the stream is coming from another predecessor node or user.

Backward traceability: Starting from an exit node of a path (or circuit), it should be possible to trace the source of a communication stream when all nodes in the path verifiably reveal their predecessors.

No forward traceability: For a compromised node, it should not be possible for the adversary $\mathcal{A}$ to use BackRef to verifiably trace its successor in any completed anonymous communication session through it.

No false accusation: It should not be possible for a compromised node to corrupt the BackRef mechanism to trace a communication stream:

1. to a path different from the path employed for the stream, and
2. to a node other than its predecessor in the path.

Non-Goals. We expect our accountability notion to be reactive in nature. We do not aim at proactive accountability and do not try to stop an illegal activity in an ACN in a proactive manner, as we believe perfect black-listing of web urls and content to be an infeasible task. Moreover, some nodes may choose not to follow the BackRef mechanism locally (*e.g.,* they may not maintain or share the required evidence logs), and full backward traceability to the user cannot be ensured in those situations; nevertheless, the cooperating nodes can still prove their innocence in a verifiable manner.

Due to its reactive nature, our repudiation mechanism inherently requires evidence logs containing verifiable routing information. Encrypting these logs and regularly rotating the corresponding keys can provide us eventual forward secrecy [48]. However, we cannot aim for *immediate* forward secrecy due to the inherently eventual forward secret nature of the encryption mechanism.

3.4.2 Design Rationale and Key Idea

Figure 3.3 presents a general expected architecture to achieve the above mentioned goals. It is clear the network level logs as well as the currently cryptographic mechanism in the ACNs cannot be used for verifiably backward traceability purpose as they cannot stop false accusations (or traceability) by compromised nodes: a compromised node can tamper with its logs to intermix two different paths as there is no cryptographic association between different parts of an ACN path.

We observe that almost all OR protocols [37, 48, 49, 50, 51, 52] (except TAP) and mix network protocols [17, 18, 38, 53, 54, 55] employ (or can employ[1]) an element of a cyclic group of prime order satisfying some (version of) Diffie-Hellman assumption as an authentication challenges or randomization element per node in the path. In particular, it can be represented as $X = g^x$, where g is a generator of a cyclic group $\mathbb{G}$ of prime order p with the security parameter κ and $x \in_R \mathbb{Z}_p$ is a random secret value known only to the user. This element is used by each node

[1]Although some of these have been defined using RSA encryptions, as discussed in [38] they can be modified to work in the discrete logarithm (DL) setting.

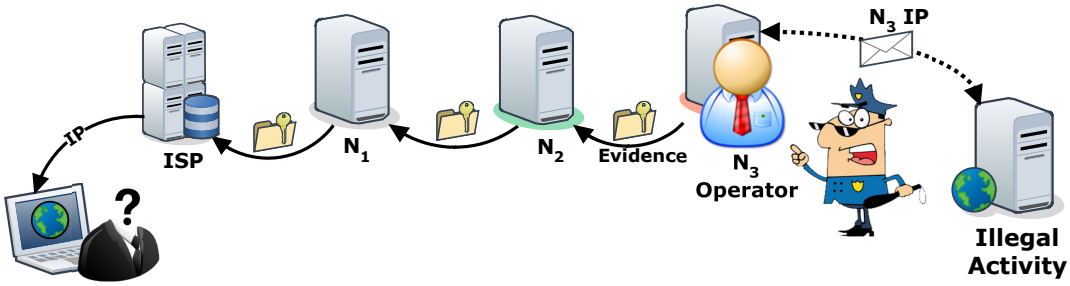


Figure 3.3: Backward traceability verification

on the path to derive a secret that is shared with the user and is used to extract a set of (session) keys for encryption and integrity protection. In the anonymity literature, these authentication challenges X are known as user *pseudonyms*.

The key idea of our BackRef mechanism is to use these pseudonyms $X = g^x$ and the corresponding secret keys x as signing key pairs to sign pseudonym's for successor nodes at entry and middle nodes, and to sign the communication stream headers at the exit nodes. Signatures that use (x, g^x) as the signing key pair are referred to as *pseudonym signatures*. As pseudonyms are generated independently for every single node, and the corresponding secret exponents are random elements of $\mathbb{Z}_p$, they do not reveal the user's identity. Moreover, it also is not possible to link two or more pseudonyms to a single identity. Therefore, pseudonym signatures become particularly useful in our BackRef mechanism, where users utilize them to sign messages without being identified by the verifier.

We can employ a CMA-secure [56] signature scheme against a computationally bounded adversary (with the security parameter κ) such that, along with the usual existential unforgeability, the resultant pseudonym signature scheme satisfies the following property:

Unconditional signer anonymity: The adversary cannot determine a signer's identity, even if it is allowed to obtain signatures on an unbounded number of messages of its choice.

We use such temporary signing key pairs (or pseudonym signatures) to sign consecutively employed pseudonyms in an ACN path and the web communication requests leaving the ACN. Pseudonym signatures provide linkability between the employed pseudonyms and the communicated message on an ACN path. However, these pseudonyms are not sufficient to link the node employed in the ACN path: for a pseudonym received by a node, its predecessor node can always deny sending the pseudonym in the first place. We solve this problem by introducing *endorsement signatures*: We assume that every node signs the pseudonym while sending it to the successor so that it cannot plausibly deny this transfer during backward tracing.

3.4.3 Scope of Solution

To understand the scope of BackRef, first consider traceability in the context of the simplest ACN: a single-hop proxy. Any traceability mechanism from the literature implicitly assumes a solution to the problem of how users can be traced through a simple proxy. We dub this the "last mile" problem. The proxy can keep logs, but this requires a trusted proxy. Alternatively the ISP could observe and log relevant details about traffic to the proxy, requiring trust in the ISP. The solution more typically used in the literature is to assume individual users have digital credentials or signing keys—essentially some form of PKI is in place to certify the keys of individual users [26, 27, 28, 29, 30].

None of these last mile solutions are particularly attractive. The assumption of a PKI provides the best distribution of trust but short-term deployment appears infeasible. We believe the involvement of ISPs is the most readily deployable. Such a solution involves an ISP with a packet attestation mechanism [57] which acts as a trusted party capable of proving the existence of a particular communication. We discuss the packet attestation mechanism further in Section 3.10.

For selected traffic flows, BackRef provides traceability to the entrance node. This is effectively equivalent to reducing the strong anonymity of a distributed cryptographic ACN to the weak anonymity of a single hop proxy. For full traceability, we then must address the "last mile" problem: tracing the flow back to the individual sender. Thus BackRef is not a full traceability mechanism, but rather an essential component that can be composed with any solution to the last mile problem. While we later discuss a solution that involves ISPs, we emphasize that BackRef itself is concentrated on, arguably, the more difficulty problem of offering ensured traceability within the ACN.

3.5 Repudiation (or Traceability)

In this section, we present our BackRef repudiation scheme. For ease of exposition, we include our scheme in an OR protocol instead of including it in the generic ACN protocol. Nevertheless, our scheme is applicable to almost all ACN mentioned in Section 3.4.2. We start our discussion with a brief overview of the OR protocol in the Tor notions [58]. We then discuss the protocol flow for BackRef, describe our cryptographic components, introduce the concept of exit node whitelisting policies, and present a formal pseudocode.

3.5.1 The OR Protocol: Overview

The OR protocol is defined in two phases: circuit construction and streams relay.

OR Circuit Construction. The circuit construction phase involves the user onion proxy (OP) randomly selecting a short circuit of (*e.g.,* 3) OR nodes, and negotiating a session key with each selected OR node using one-way authenticated key exchange (1W-AKE) [52] such as the ntor protocol. (Section 3.9 presents more details for the 1W-AKE protocol.) When a user wants to create a circuit with an OR node N_1, she runs the *Initiate* procedure of the ntor protocol to generate and send an authentication challenge to N_1. Node N_1 then runs the *Respond* procedure and returns the authentication response. Finally, the user uses the *ComputeKey* procedure of ntor along with the response to authenticate N_1 and to compute a session key with it. To extend the circuit further, the user sends an extend request to N_1 specifying the address of the next node N_2 and a new ntor authentication challenge for N_2. The process continues to until the user exchanges the key with the exit node N_3.

Relaying Streams. Once a circuit (denoted as $\langle U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3 \rangle$) has been constructed through N_1, N_2 and N_3, the user-client U routes traffic through the circuit using onion-wrapping *WrOn* and onion-unwrapping *UnwrOn* procedures. *WrOn* creates a layered encryption of a payload (plaintext or onion) given an ordered list of (three) session keys. *UnwrOn* removes one or more layers of encryptions from an onion to output a plaintext or an onion given an input onion and a ordered list of one or more session keys. To reduce latency, many of the user's communication streams employ the same circuit [34].

The structure and components of communication streams may vary with the network protocol. For ease of exposition, we assume the OR network uses TCP-based communication in the same way as Tor, but our schemes can easily be adapted for other types of communication streams.

In Tor, the communication between the user's TCP-based application and her Tor proxy takes place via SOCKS. To open a communication stream (*i.e.,* to start a TCP connection to some web server and port), the user proxy sends a *relay begin* cell (or packet) over the circuit to the exit node N_3. When N_3 receives the TCP request, it makes a standard TCP handshake with the web server. Once the connection is established, N_3 responds to the user with a *relay connected* cell. The user then forwards all TCP stream requests for the server as *relay data* cells to the circuit. (See [34, 58] for a detailed explanation.)

3.5.2 The BackRef Protocol Flow

Consider a user U who wishes to construct an OR circuit $\langle U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3 \rangle$, and use it to send communication stream m. BackRef adds the repudiation mechanism as a layer on the top of the existing OR protocol. We assume that every OR node possesses a signing (private) key for which the corresponding verification (public) key is publicly available through the OR directory service.

The corresponding OR protocol with the BackRef scheme works according to the following five steps:

1. Circuit construction with an entry node. The user U creates a circuit with the entry node N_1 using the ntor protocol. If the user is an OR node, then it endorses its pseudonym X_1 by signing it with its public key and sending the signature along with X_1.

However, if the user U is not an OR node, it cannot endorse the pseudonym X_1 as no public-key infrastructure (PKI) or credential system is available to him. We solve this endorsement problem by entrusting the ISP with a packet attestation mechanism [57] such that the ISP can prove that a pseudonym was sent from U to N_1. We discuss the packet attestation mechanism in Section 3.10.

2. Circuit extension. To extend a circuit to N_2, U generates a new pseudonym X_2 of an ntor instance, signs X_2 and the current timestamp with the secret value x_1 associated with X_1, and sends an extend request to N_1 along with the identifier for N_2, $\{X_2||\text{ts}_{x_2}\}_{\sigma_{X_1}}$ and a timestamp ts_{x_2}. Notice that the extension request is encrypted by a symmetric session key negotiated between U and N_1.

Upon receiving a message, N_1 decrypts and verifies $\{X_2||\text{ts}_{x_2}\}_{\sigma_{X_1}}$ using the previously received pseudonym X_1 and timestamp. We call this verification *pseudonyms linkability verification*. If the signature is valid, it creates an evidence record as discussed in Step 4, signs X_2 using its private key to generate $\{X_2||\text{ts}_2\}_{\sigma_{sk_2}}$ and sends a circuit create request to the node N_2 with $\{X_2||\text{ts}_2\}_{\sigma_{sk_2}}$.

Node N_2, upon receiving a circuit creation request along with $\{X_2||ts_2\}_{\sigma_{sk_2}}$, verifies the signature. Upon a successful verification, it replies to N_1 with an ntor authentication response for the OR key agreement and generates the OR session key for its session with (unknown) user U. N_1 sends the authentication response back to U using their OR session, who then computes the session key with N_2 and continues to build its circuit to N_3 in a similar fashion.

Notice that we carefully avoid any conceptual modification of the OR circuit construction protocol; the above signature generation and verification steps are the only adjustments that BackRef makes to this protocol.

3. Stream verification. Once a circuit $\langle U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3 \rangle$ has been established, the user U can utilize it to send her web stream requests. To open a TCP connection, the user sends a *relay begin* cell to the exit node N_3 through the circuit. The user U includes a pseudonym signature (or stream request signature) on the cell contents signed with the secret exponent x_3 of X_3. The user also includes a timestamp in her stream request. When the *relay* cell reaches the exit node N_3, the exit node verifies the pseudonym signature with X_3. Once the verification is successful and the timestamp is current, N_3 creates the evidence log (Step 4) and proceeds with the TCP handshake to the destination server. The *relay* stream request is discarded otherwise.

This stream verification helps N_3 to prove linkability between its handshakes

with the destination server and the pseudonym X_3 it received from N_2. When a whitelist directory exists, the exit node first consults the directory and if the request (*i.e.,* web stream request) is whitelisted, the exit node just forwards it to the destination server. In such a case, the exit node does not require any signature verification and also does not create an evidence log. We further discuss the server whitelisting in Section 3.5.4.

4. Log generation. After every successful pseudonym linkability or stream verification, the evidence record is created. A pseudonym linkability verification evidence record associates linkability between two pseudonyms X_i and X_{i+1} and an endorsement signature on X_i, while a stream verification evidence record associates a stream verification with an endorsement signature on X_3 for N_3.

5. Repudiation or traceability. The verifier contacts the exit node N_3 with the request information (*e.g.,* IP address, port number, and timestamp) for a malicious stream coming out of the exit node N_3. The operator of N_3 can determine a record using the stream request information. This evidence record verifiably reveals the identity of the middle node N_2.
As an optional next step, using the evidence records, it is possible for N_2 to verifiably reveal the identity of the predecessor node N_1. Then, the last mile of a full traceability is to reach from N_1 to the user U in a verifiable manner using the evidence record on N_1 and the request information on the ISP [57]. When the user U is an OR node a record at N_1 is sufficient and the last mile problem does not exist.

3.5.3 Cryptographic Details

For pseudonym and endorsement signatures, we use the short signature scheme of Boneh, Lynn and Shacham (BLS) [59]. We recall the BLS signature scheme in Section 3.7. We choose the BLS signature scheme due to the shorter size of their signatures; however, if signing and verification efficiency is more important, we can choose faster signature schemes such as [60].

Circuit Extension. To extend the circuit $\langle U \leftrightarrow N_1 \rangle$ to the next hop N_2, the user U chooses $x_2 \in_R \mathbb{Z}_p$ and generates a pseudonym $X_2 = g_2^{x_2}$, where $g_2 \in \mathbb{G}_2$. U then signs the pseudonym X_2 with pseudonym X_1 as public key. Also we include the current timestamp value ts_{x_2} in the signature $\sigma_{X_1} = H(X_2||\text{ts}_{x_2})^{x_1}$. Upon receiving the signed pseudonym $\{X_2||\text{ts}_{x_2}\}_{\sigma_{X_1}}$ along with the timestamp ts_{x_2}, the node N_1 checks if the timestamp is current and verifies it as follows:

$$e(H(X_2||\text{ts}_{x_2}), X_1) \stackrel{?}{=} e(\sigma_{X_1}, g_2)$$

Pseudonym endorsement. After successful verification, N_1 creates an endorsement signature $\sigma_1 = H(X_2||\text{ts}_2)^{sk_1}$ for pseudonym X_2 and current timestamp

ts_2 using its signing key sk_1 and sends it along with X_2 and ts_2 to N_2. The node N_2 then follows the pseudonym endorsement step. Upon receiving the signed pseudonym $\{X_2||\text{ts}_2\}_{\sigma_1}$, the exit node N_2 verifies it as follows:

$$e(H(X_2||\text{ts}_2), pk_1) \stackrel{?}{=} e(\sigma_1, g_2).$$

On a successful verification, N_2 continues with the OR protocol.

Stream verification. To generate a stream request signature, the user signs the stream request (*i.e.,* selected contents of the *relay begin* cell) using the pseudonym $X_3 = g_2^{x_3}$ where x_3 is the secret corresponding to X_3. For contents of the *relay* cell $m = \{\text{address}||\text{port}||\text{ts}_{x_m}\}$, the stream request signature σ_{X_3} is defined as

$$\sigma_{X_3} = H(m)^{x_3}.$$

The user sends the signature along with the *relay* cell and the current timestamp ts_{x_m} to the exit node through the already-built circuit.

Once the signed stream request reaches N_3, it verifies the signature as follows:

$$e(H(m), X_3) \stackrel{?}{=} e(\sigma_{X_3}, g_2). \tag{3.1}$$

Upon a successful verification, the exit node N_3 proceeds with the TCP handshake. A verified request allows the node to link X_3 and the request.

Log generation. After every successful pseudonym or stream verification, an evidence record is added to the evidence log. The evidence records differ with the position of the nodes within a circuit, and we define two types of evidence logs.

Exit node log: For every successful stream verification, an evidence record is added to the evidence log at the exit node. A single evidence record consists of the signature on X_3 (*i.e.,* $\{X_3||\text{ts}_3\}_{\sigma_2}$), and the stream request ($m = \{\text{address}||\text{port}||\text{ts}_{x_m}\}$) coupled by the pseudonym signature $\{m\}_{\sigma_{X_3}}$ and the timestamp ts_{x_m}.

Middle and entry node log: The middle and entry node evidence record comprises two pseudonyms X_i, X_{i+1}, and a timestamp value $\text{ts}_{x_{i+1}}$ coupled with the appropriate signatures and the IP address of N_{i-1}. The pseudonym X_i is coupled with an endorsement signature $\{X_i||\text{ts}_i\}_{\sigma_{i-1}}$ from node N_{i-1}, and the pseudonym X_{i+1} is coupled by a pseudonym signature $\{X_{i+1}||\text{ts}_{x+1}\}_{\sigma_{X_i}}$. When the user is not an OR node and does not posse a verifiable signature key pair, the corresponding record at N_1 consists of a signed pseudonym $\{X_2||\text{ts}_{x_2}\}_{\sigma_{X_1}}$, pseudonym X_1, timestamp value ts_{x_2}, and the IP of the user.

Repudiation or traceability. Given the server logs of a stream request, an evidence record corresponding to the stream request can be obtained. In the first step, it is checked whether the timestamp matches the stream request under observation. In the next step, the association between the stream request and the pseudonym of the exit node X_3 is verified using the pseudonym signature. Then, the association of the pseudonym X_3 and N_2 is checked using the pseudonym endorsement signature.

Given the pseudonym X_3 and a timestamp ts_{x_m}, the backward traceability verification at node N_2 is carried out as follows:

1. Do a lookup in the evidence log to locate the signed pseudonym $\{X_3||\text{ts}_{x_3}\}_{\sigma_{X_2}}$ and the timestamp ts_{x_3}, where X_3 is the lookup index.

2. Compare the timestamps (ts_{x_m} and ts_{x_3}) under observation and prove the linkability between X_2 and X_3 by verifying the signature $\{X_3||\text{ts}_{x_3}\}_{\sigma_{X_2}}$.

3. If verification succeeds, reveal the IP address of the node N_1 who has forwarded X_2 and verify $\{X_2||\text{ts}_2\}_{\sigma_1}$ with pk_1.

The above three steps can be used repeatedly to reach the entry node. However, they cannot be used to verifiably reach the user if we do not assume any public key and credential infrastructure for the users. Instead, our protocol relies on the ISP between user U and N_1 to use packet attestation [57] to prove that the pseudonym X_1 was sent from U to N_1.

3.5.4 Exit Node Whitelisting Policies

To provide a good balance between anonymity and accountability, we include a whitelisting option for exit nodes. This option allows a user to avoid the complete verification and logging mechanisms if her destination is in the whitelist directory of her exit node. In particular, we categorize the destinations into two groups:

Whitelisted destinations. For several destinations such as educational `.edu` websites, an exit node may find traceability to be unnecessary. The exit node includes such destinations in a whitelist directory such that, for these destinations, the employed circuit nodes do not demand any endorsement and pseudonym signatures. Traffic sent to these whitelisted destinations through the circuit remains anonymous in the current ACN sense.

Non-listed destinations. For destinations that are not listed in the exit-node whitelist directory, the user has to use BackRef while building the circuit to it; otherwise, the exit node will drop her requests to the non-listed destinations. We emphasize that BackRef is *not* an "all-or-nothing" design alternative: it allows

an ACN to conveniently disable the complete verification and logging mechanisms for some pre-selected destinations. In particular, an exit node with "Sorry, it is an anonymity network, no logs" opinion can still whitelist the whole Internet, while others employ BackRef for non-whitelisted sites. The use of BackRef is transparent, and users can choose if they wish to use a BackRef node for their circuits.

3.5.5 Pseudocode

In this subsection, we present pseudocode for the OR protocol with BackRef extending the OR pseudocode developed by Backes *et al.* [61] following the Tor specification [58]. In Listing 3.1, we highlight our changes to their original (Π_{OR}) protocol pseudocode from [61] by underlining those. Our pseudocode formalism demonstrates that our modifications of the original OR protocol are minimal. It also forms the basis for our applied pi calculus [62] based OR model in Section 3.6. In the pseudocode, an OR node maintains a state for every protocol execution and responds (changes the state and/or sends a message) upon receiving a message.

There are two types of messages that the protocol employs: the first type contains *input* and *output* actions, which carry respectively the user inputs to the protocol, and the protocol outputs to the user; the second message type is a network message (a cell in the OR literature), which is to be delivered by one protocol node to another. In Listing 3.2, we formalize the backward traceability verification of BackRef. Here, function *LookupLog* determines an entry from the log index by its input. Function $Verify$ performs signature verification, while function $TraceFail$ outputs that a valid log entry does not exists at node N.

In onion routing, a directory server maintains the list of valid OR nodes and the respective public keys. A functionality $\mathcal{F}_{\text{REG}}^{\mathcal{N}}$ abstracts this directory server. Each OR node initially computes its long-term keys (sk, pk) (for both 1W-AKE and signature schemes) and registers the public part at $\mathcal{F}_{\text{REG}}^{\mathcal{N}}$. For ease of exposition, cryptographically important Tor cells are considered in the protocol. This includes create, created and destroy cells among control cells, and data, extend and extended cells among relay cells. There are two input messages createcircuit and send, where the user uses createcircuit to create OR circuits and uses send to send messages m over already-created circuits.

The *ExtendCircuit* function defined in Listing 3.3 presents the circuit construction description from Section 3.5.1 in a pseudocode form. Circuit IDs ($cid \in \{0,1\}^{\kappa}$) associate two consecutive circuit nodes in a circuit. The terminology $\mathcal{C} = N_{i-1} \overset{cid_i,k_i}{\Longleftrightarrow} N_i \overset{cid_{i+1}}{\Longleftrightarrow} N_{i+1}$, says that N_{i-1} and N_{i+1} are respectively the predecessor and successor of N_i in a circuit $\mathcal{C}$. k_i is a session key between N_i and the OP, while the absence of k_{i+1} indicates that a session key between N_{i+1} and the OP is not known to N_i; analogously the absence of a circuit id cid in that notation means that only the first circuit id is known, as for OP, for example.

Functions *prev* and *next* on *cid* correspondingly return information about the predecessor or successor of the current node with respect to *cid*; *e.g.,* $next(cid_i)$ returns (N_{i+1}, cid_{i+1}) and $next(cid_{i+1})$ returns $\perp$. The OP passes on to the user $\langle N \overset{cid_1}{\Longleftrightarrow} N_1 \Longleftrightarrow \cdots N_\ell \rangle$.

Within a circuit, a user's OP (onion proxy) and the exit node use relay cells created using wrapping algorithm *WrOn* to tunnel commands and communication. The exit nodes use the streams to synchronize communication between the network and a circuit $\mathcal{C}$. It is represented as *sid* in the pseudocode. End-to-end communication between OP and the exit node happens with a *WrOn* call with multiple session keys and a series of *UnwrOn* calls with individual session keys. Cells are exchanged between OR nodes over a secure and authenticated channels $\mathcal{F}_{\text{SCS}}$ [63]. Circuit destruction remains the same so we *omit* it in our pseudocode and refer the readers to [61] for more details.

Listing 3.1 Π_{OR} with BackRef for party N (without circuit destruction)

upon an input (setup):
  Generate an asymmetric key pair $(sk, pk) \leftarrow G$.
  send a cell $(\mathsf{register}, N, pk)$ to the $\mathcal{F}_{\text{REG}}^{\mathcal{N}}$ functionality
  wait for a cell $(\mathsf{registered}, \langle N_j, pk_j \rangle_{j=1}^{n})$ from $\mathcal{F}_{\text{REG}}^{\mathcal{N}}$
  output $(\mathsf{ready}, \mathcal{N} = \langle N_j \rangle_{j=1}^{n})$
upon an input $(\mathsf{createcircuit}, \mathcal{N} = \langle N, \langle N_j \rangle_{j=1}^{\ell} \rangle)$:
  store $\mathcal{N}$ and $\mathcal{C} \leftarrow \langle N \rangle$; call $ExtendCircuit(\mathcal{N}, \mathcal{C})$
upon an input $(\mathsf{send}, \mathcal{C} = \langle N \overset{cid_1}{\Longleftrightarrow} N_1 \Longleftrightarrow \cdots N_\ell \rangle, m)$:
  look up the keys $(\langle k_j \rangle_{j=1}^{\ell})$ for cid_1
  $O \leftarrow WrOn(m, \sigma_{X_\ell}, \text{ts}, (k_j)_{j=1}^{\ell})$; $Used(cid_1)$++
  send a cell $(cid_1, \overline{\mathsf{relay}}, O)$ to N_1 over $\mathcal{F}_{\text{SCS}}$
upon receiving a cell $(cid, \mathsf{create}, X, \underline{\sigma_i, \text{ts}})$ from N_i over $\mathcal{F}_{\text{SCS}}$:
  if $\overline{Verify(\sigma_i, pk_{N_i})}$ **then**
    $\langle Y, k_{\mathsf{new}} \rangle \leftarrow Respond(pk_N, sk_N, X)$
    store $\mathcal{C} \leftarrow \langle N_i \overset{cid, k_{\mathsf{new}}}{\Longleftrightarrow} N \rangle$
    store $\underline{Log \leftarrow \langle H(X), IP_{N_i} X, \sigma_i, \text{ts} \rangle}$
    $\overline{\text{send a cell } (cid, \mathsf{created}, Y, t) \text{ to } N_i \text{ over } \mathcal{F}_{\text{SCS}}}$
upon receiving a cell $(cid, \mathsf{created}, Y, t)$ from N_i over $\mathcal{F}_{\text{SCS}}$:
  if $prev(cid) = (N', cid', k')$ **then**
    $O \leftarrow WrOn(\langle \mathsf{extended}, Y, t \rangle, k')$
    send a cell $(cid', \mathsf{relay}, O)$ to N' over $\mathcal{F}_{\text{SCS}}$
  else if $prev(cid) = \perp$ **then**
    $k_{\mathsf{new}} \leftarrow ComputeKey(pk_i, Y, t)$
    update $\mathcal{C}$ with k_{new}; call $ExtendCircuit(\mathcal{N}, \mathcal{C})$

upon receiving a cell (cid, relay, O) from N_i over $\mathcal{F}_{\text{SCS}}$:
  if $prev(cid) = \perp$ **then**
    if $getkey(cid) = (k_j)_{j=1}^{\ell'}$ **then**
      (type, m) **or** $O \leftarrow UnwrOn(O, (k_j)_{j=1}^{\ell'})$
      (N', cid') **or** $\perp \leftarrow next(cid)$
  else if $prev(cid) = (N', cid', k')$ **then**
    $O \leftarrow WrOn(O, k')$ /* a backward onion */
  switch (type)
  case extend:
    get $\langle N_{next}, X, \sigma_{X_i}, \text{ts}\rangle$ from m; $cid_{next} \xleftarrow{\$} \{0,1\}^\kappa$
  if $Verify(\sigma_{X_i}, X_i)$ **then**
      update $\mathcal{C} \leftarrow \langle N_i \overset{cid,k}{\Longleftrightarrow} N \overset{cid_{next}}{\Longleftrightarrow} N_{next}\rangle$
      store $Log \leftarrow \langle H(X), IP_{N_i} X, \sigma_{X_i}, \text{ts}\rangle$
      send a cell $(cid_{next}, \mathsf{create}, X)$ to N_{next} over $\mathcal{F}_{\text{SCS}}$
  case extended:
    get $\langle Y, t\rangle$ from m; get N_{ex} from $(\mathcal{C}, \mathcal{N})$
    $k_{\mathsf{ex}} \leftarrow ComputeKey(pk_{\mathsf{ex}}, Y, t)$
    update $\mathcal{C}$ with (k_{ex}); call $ExtendCircuit(\mathcal{N}, \mathcal{C})$
  case data:
    if (N = OP) **then** output $(\mathsf{received}, \mathcal{C}, m)$
    else if $m = (S, m', \sigma_X, \text{ts})$
      store $Log \leftarrow \langle H(m), IP_{N_i}, X, \sigma_X, \text{ts}\rangle$
      generate or lookup the unique sid for cid
      send (N, S, sid, m') to the network
  case default: /*encrypted forward/backward onion*/
    send a cell $(cid', \mathsf{relay}, O)$ to N' over $\mathcal{F}_{\text{SCS}}$
upon receiving a msg (sid, m) from $\mathcal{F}_{\text{NET}^q}$:
  get $\mathcal{C} \leftarrow \langle N' \overset{cid,k}{\Longleftrightarrow} N\rangle$ for sid; $O \leftarrow WrOn(m, k)$
  send a cell (cid, relay, O) to N' over $\mathcal{F}_{\text{SCS}}$

3.6 Security Analysis

In this section we present a formal security analysis of BackRef. We model our protocol from the previous section (in a restricted form) in the applied pi calculus [62] and verify the important properties anonymity, backward traceability, no forward traceability, and no false accusation with ProVerif [64], a state-of-the-art automated theorem prover that provides security guarantees for an unbounded number of protocol sessions. We model backward traceability and no false accusation as trace properties, and anonymity and no forward traceability as observational equivalence relations. The ProVerif scripts used in the analyses

Listing 3.2 Backward Traceability Verification

upon a verification request (m):
  if $LookupLog(H(m)) = \perp$ **then**
    $TraceFail(m)$
  else
    get $Log \leftarrow \langle H(m), N_{prev}, X, \sigma, \mathrm{ts}\rangle$ for $H(m)$
    if $((\mathrm{N} = \mathrm{N}_1)$ & $Verify(\sigma, X))$ **then**
      output (X, N_{prev})
    else
      get $Log \leftarrow \langle H(X), N_{N_{prev}}, pk_{N_{prev}}, \sigma', \mathrm{ts}\rangle$ for $H(X)$
      if $(Verify(\sigma, X)$ & $Verify(\sigma', pk_{N_{prev}}))$ **then**
        output (X, N_{prev})
      else
        $TraceFail(m)$

are publicly available [65].

In Listing 3.2, we formalize the backward traceability verification of BackRef. Here, function *LookupLog* determines an entry from the log index by its input. Function $Verify$ performs signature verification, while function $TraceFail$ outputs that a valid log entry does not exists at node N.

Listing 3.3 Subroutine for Π_{OR} with BackRef for N

ExtendCircuit$(\mathcal{N} = \langle N_j\rangle_{j=1}^{\ell}, \mathcal{C} = \langle N \overset{cid_1,k_1}{\Longleftrightarrow} N_1 \overset{k_2}{\Longleftrightarrow} \cdots N_{\ell'}\rangle)$:
  determine the next node $N_{\ell'+1}$ from $\mathcal{N}$ and $\mathcal{C}$
  if $N_{\ell'+1} = \perp$ **then**
    output $(\mathsf{created}, \langle N \overset{cid_1}{\Longleftrightarrow} N_1 \Longleftrightarrow \cdots N_{\ell'}\rangle)$
  else
    $X \leftarrow Initiate(pk_{N_{\ell'+1}}, N_{\ell'+1})$
    if $N_{\ell'+1} = N_1$ **then**
      $cid_1 \overset{\$}{\leftarrow} \{0,1\}^{\kappa}$
      send a cell $(cid_1, \mathsf{create}, X)$ to N_1 over $\mathcal{F}_{\mathrm{SCS}}$
    else
      $O \leftarrow WrOn(\{\mathsf{extend}, N_{\ell'+1}, X, \sigma_{X_{\ell'}}, \mathrm{ts}\}, (k_j)_{j=1}^{\ell'})$
      send a cell $(cid_1, \mathsf{relay}, O)$ to N_1 over $\mathcal{F}_{\mathrm{SCS}}$

Basic Model. We model the OR protocol in the applied pi calculus to use circuits of length three (*i.e.,* one user and three nodes); the extension to additional nodes is straightforward. To prove different security properties we upgrade the model to use additional processes and events. The event contents used to decorate

the various steps in the OR protocol as well as BackRef mechanism follow the pseudocode from the previous section. We also involve an ISP between the user and the entry node, which participate in the protocol as a trusted party. The ISP is honest and can prove the existence of a communication channel between the user and the entry node. This channel is modeled to be private, preventing any ISP log forgeries. The cryptographic log collection model is designed in a decentralized way such that nodes retain the logs themselves in a table that is inaccessible to the adversary.

We model the flow of the pseudonyms and the onion, together with the corresponding verification. However, we do not model the underlying, cryptographically verified 1W-AKE ntor protocol, and assume that the session key between the user and the selected OR process is exchanged securely. The attacker is a standard Dolev-Yao active adversary with full control over the public channels: It learns everything ever sent on the network, and can create and insert messages on the public channels. It also controls network scheduling.

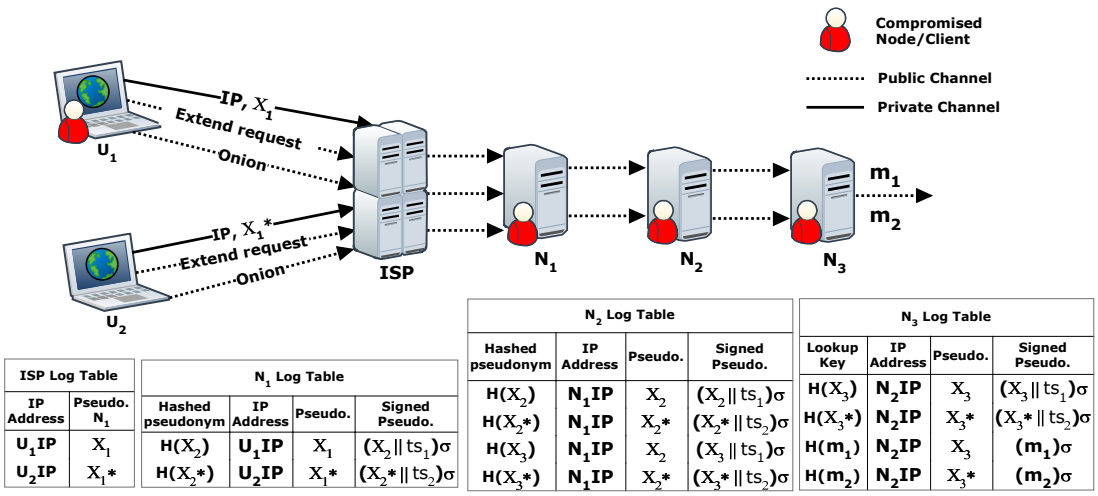
Backward Traceability. The essential goal of our protocol is to trace the source of the communication stream starting from an exit node. We verify that the property of backward traceability arrives from the correctness of the (backward) traceability verification mechanism.

The correctness property can be formalized in ProVerif notation as follows:

$$\begin{aligned} TraceUser(IP) \Longrightarrow (LookupISP(X_1, IP) \Longrightarrow \\ (RevealPredecessorU(IP)) \Longrightarrow \\ (RevealPredecessor(ipN1)) \Longrightarrow \\ (RevealPredecessor(ipN2)) \wedge CheckSignature \\ \wedge LookupN3(m))) \end{aligned} \tag{3.2}$$

where the notation A $\Longrightarrow$ B denotes the requirement that the event A must be preceded by a event B. In our protocol, the property says that the user is traced if and only if all nodes in the circuit verifiably trace their predecessors. The traceability protocol P starts with the event $LookupN3(m)$ which means that for a given message m (stream request) the verifier consults the log, and if such request exists, it checks the signature $CheckSignature$. Finally when all these conditions are fulfilled, the verifier reveals the identity of the predecessor node $RevialPredecessor(ipN2)$ (*i.e.,* the middle node). This completes the nested correspondence $(CheckSignature \wedge LookupN3(m) \wedge RevealPredecessor(ipN2))$ which verifiably traces N_2. In a similar fashion, after all conditions are fulfilled, the verifier traces N_1 and the user U.

After the identity of U is revealed, the verifier lookup into the evidence table of the ISP ($LookupISP$) to prove the connection between the identity of the user



ISP Log Table

IP Address	Pseudo. N_1
U_1IP	X_1
U_2IP	X_1*

N_1 Log Table

Hashed pseudonym	IP Address	Pseudo.	Signed Pseudo.
$H(X_2)$	N_1IP	X_2	$(X_2 \| ts_1)\sigma$
$H(X_2*)$	N_1IP	X_2*	$(X_2* \| ts_2)\sigma$
$H(X_3)$	N_1IP	X_2	$(X_3 \| ts_1)\sigma$
$H(X_3*)$	N_1IP	X_2*	$(X_3* \| ts_2)\sigma$

N_2 Log Table

Lookup Key	IP Address	Pseudo.	Signed Pseudo.
$H(X_3)$	N_2IP	X_3	$(X_3 \| ts_1)\sigma$
$H(X_3*)$	N_2IP	X_3*	$(X_3* \| ts_2)\sigma$
$H(m_1)$	N_2IP	X_3	$(m_1)\sigma$
$H(m_2)$	N_2IP	X_3*	$(m_2)\sigma$

Figure 3.4: No false accusation adversarial model

IP and the pseudonym of the entry node X_1. If such record exist into the table, the address of the user is revealed and the event $TraceUser(IP)$ is executed.

Theorem 1. *The trace property defined in Equation (3.2) holds true for all possible executions of process P.*

Proof. Automatically proven by ProVerif. □

No false accusation. There are two aspects associated with false accusations:

1. It should not be possible for a malicious node N_A to trace a communication stream to an OR node N_C other than to its predecessor in the corresponding circuit. Informally, to break this property, N_A has to be obtain a signature of N_C on a particular pseudonym associated with the circuit. This requires N_A to forge a signature for N_C, which is not possible due to the unforgeability property of the signature scheme.

2. It should not be possible for a malicious node N_A to trace a communication stream to a circuit C_1 other than the circuit C_2 employed for the communication stream. Consider a scenario where two concurrent circuits (C_1 and C_2), established by two different users U_1 and U_2, pass through a malicious node N_A. Suppose that N_A collaborates with U_2 who is misbehaving and have used the OR network for a criminal activities. To help U_2 by falsely accusing a different predecessor, N_A must forge two signatures: To link two pseudonyms X_{1i-1} and X_{2i} from circuits C_1 and C_2 respectively, N_A has to forge the pseudonym signature on X_{2i} with X_{1i-1} as a public key, or he has to know the temporal signing key pair for the predecessor in C_1.

Intuitively, the first case is ruled out by the unforgeability property of the signature scheme. We model the later case as a trace property. Here, even when N_A collaborates with U_2, it cannot forge the signed pseudonym received from its predecessor. The property remains intact as long as one of nodes on C_1 and the packet attesting ISP [57] remains uncompromised. In absence of a PKI or credential system for users, the last condition is unavoidable.

We formalize and verify the latter case of the property in an adversarial model where the attacker has compromised one user (U_1 or U_2). Figure 3.4 provides a graphical representation of the protocol P. We upgrade the basic model involving additional user U_2 who sends additional message m_2. As mentioned before, to simulate the packet attesting mechanism [57] we involve a honest ISP between the user and the entry node. The ISP only collects data that identifies the user (IP address of the user) and the pseudonym for the entry node (X_1) which is send in plain-text. The adversary does not have an access to the log stored by the ISP, *i.e.,* cannot read or write anything into the log table. We want to verify that for all protocol executions the request m_i cannot be associated with any user U_i other than the originator.

To formalize the no false accusation property in ProVerif, we model security-related protocol events with logical predicates. The event CorrISP defines the point of the protocol where the ISP is corrupted. The no false accusation property is formalize as the following policy:

$$Accuse(IP, m) \Longrightarrow CorrISP. \tag{3.3}$$

This policy says that if a user with address IP is falsely accused for a message m, *i.e.,* $Accuse(IP, m)$, then indeed the ISP have to be corrupted.

Theorem 2. *The trace property defined in Equation (3.3) holds true for all possible executions of process P.*

Proof. Automatically proven by ProVerif. □

Anonymity. We model this property as an observational equivalence relation between two processes that are replicated an unbounded numbers of time and execute in parallel. In the first process P, users U_1 and U_2 send two messages m_1 and m_2, respectively. While in the second process Q the two messages are swapped. If the two defined processes are observationally equivalent (P $\approx$ Q), then we say that the attacker cannot distinguish between m_1 and m_2, *i.e.,* cannot learn which message is sent by which user. In our scenario we assume that the attacker can compromise some fraction of the OR node, but not all. Figure 3.5 provide a graphical representation of the anonymity game where the exit node N_3 is honest. The game works as follows:

1. U_1 and U_2 create an onion data structure O_1 and O_2, respectively, intended for N_3 and send via previously built circuits C_1 ($U_1 \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3$) and $C_2(U_2 \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3)$. Nodes communicate between each other through public channel.

2. Two of the intermediate nodes are corrupted and the attacker has full control over them. The intermediate compromised nodes (in our case N_1 and N_2) remove one layer of encryption from O_1 and O_2 and send the onion to the exit node N_3.

3. After receiving these two onions from the users U_1 and U_2 and possibly other onions from compromised users, the exit OR node N_3 remove the last layer of the encryption and publish the message on a public channel.

Note that the ISP does not affect the anonymity game and only acts as a proxy between the users and the outside world. For the anonymity verification, we assume that user U_1 and user U_2 are honest and they follow the protocol. Nevertheless, the action of any compromised user and honest users can be interleaved in any order.

Theorem 3. *The observational equivalence relation $P \approx Q$ holds true.*

Proof. Automatically proven by ProVerif. □

Notice that the evidence records here inherently break anonymity: anybody with access to logs of the entry, middle, and exit nodes of a circuit can break the user anonymity. Therefore, traceability logs have to be indexed and individually encrypted using an appropriate trust-enforcing mechanism. In Section 3.10, we discuss the possible solutions.

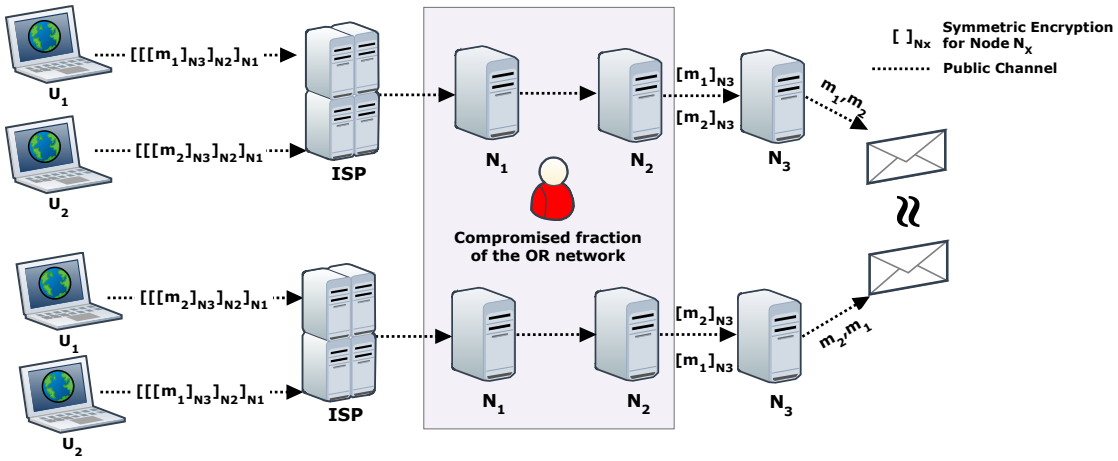


Figure 3.5: Anonymity game

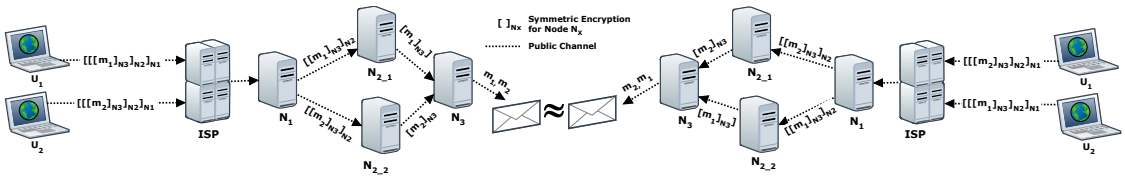


Figure 3.6: No forward traceability

No forward traceability. The evidence log of the backward traceability protocol in BACKREF does not store any information (*i.e.,* IP addresses) that can identify or verifiably reveal the identity of a node's successor. The log contains only the pseudonym for the successor node which does not reveal anything about the identity of the node.

We formalize this property as an observational equivalence relation between two distinct processes and verify that an adversary cannot distinguish them. Figure 3.6 provides a graphical representation of the game. To prove the observational equivalence, we model a scenario with concurrent circuit executions. In this game, the adversary can corrupt parties and extract their secrets only after the message transmission over the circuit has completed. For this game, our model involves an additional middle node and user U_2. Two users U_1 and U_2 send two different messages m_1 and m_2 via two circuits. We verify that it is impossible for an attacker to deduce any meaningful information about the successor node for a particular request. Our game works as follows:

1. U_1 and U_2 start the protocol and constructs two different circuits $C_1(U \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow N_3)$ and $C_2(U \leftrightarrow N_1 \leftrightarrow N_2^* \leftrightarrow N_3)$, respectively with adequate values (x_1, x_2, x_3) for a circuit C_1 and (x_1', x_2', x_3') for C_2.

2. U_1 and U_2 create an onion data structure O_1 and O_2 and send to the exit node N_3 via previously built circuits C_1 and C_2. Nodes communicate between each other through public channels.

3. After receiving the two onions from the users and possibly other onions from compromised users, N_3 removes the last layer of the encryption and publishes the messages on a public channel.

4. After protocol completion, the entry node N_1 is compromised and the adversary obtains the evidence log.

In the first process P, U_1 sends m_1 and U_2 sends m_2, while the process Q is reversed process P. For the no forward traceability verification, we assume that all other parties in the protocol remain honest, except the compromised N_1. For example, if two neighbor nodes are compromised, the no forward traceability can be easily broken with activating the backward traceability mechanism.

Theorem 4. *The observational equivalence relation $P \approx Q$ holds true.*

Proof. Automatically proven by ProVerif. □

Finally, to the best of our knowledge, our formal analysis is the first ProVerif-based analysis of the OR protocol; it can be of independent interest towards formalizing and verifying other properties of the OR protocol.

3.7 BLS Signatures

In this section, we briefly review BLS signatures. For more details see [59] and references therein.

Consider two Gap co-Diffie-Hellman groups (or co-GDH group) $\mathbb{G}_1$ and $\mathbb{G}_2$ and a multiplicative cyclic group $\mathbb{G}_T$, all of the same prime order p, associated by a bilinear map [66] $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. (We include a small note on bilinear maps in Section 3.8.) Let g_1, g_2, and g_T be generators for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ respectively and let a full-domain hash function $H : \{0,1\}^* \to \mathbb{G}_1$. The BLS signature scheme [59] comprises three algorithms, *Key Generation*, *Signing* and *Verification* defined as follows:

Key Generation: Choose random $sk \in_R \mathbb{Z}_p$ and compute $pk = g_2^{sk}$. The private key is sk, and the public key is pk.

Signing: Given a private key $pk \in \mathbb{Z}_p$, and a message $m \in \{0,1\}^*$, compute $h = H(m) \in \mathbb{G}_1$ and signature $\sigma = h^{sk}$, where $\sigma \in \mathbb{G}_1$.

Verification: Given a public key $pk \in \mathbb{G}_2$, message $m \in \{0,1\}^*$, and signature $\sigma \in \mathbb{G}_1$, compute $h = H(m) \in \mathbb{G}_1$ and verify that (g_2, pk, h, σ) is a valid co-Diffie-Hellman tuple.

3.8 Bilinear Pairings

In this section, we briefly review bilinear pairings. For more details see [66] and references therein.

Consider two additive cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and a multiplicative cyclic group $\mathbb{G}_T$, all of the same prime order p. A bilinear map e is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties.

Bilinearity: For all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(P^a, Q^b) = e(P, Q)^{ab}$.

Non-degeneracy: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ to unity in $\mathbb{G}_T$.

Computability: There is an efficient algorithm to compute $e(P, Q)$ for any $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.

3.9 1W-AKE Protocol

Until recently, Tor has been using an authenticated Diffie-Hellman (DH) key agreement protocol called the Tor authentication protocol (TAP), where users' authentication challenges are encrypted with RSA public keys of OR nodes. However, this atypical use of RSA encryption is found to be inefficient in practice, and several different interactive and non-interactive (one-way authenticated) key agreement (1W-AKE) protocols have been proposed in the literature [48, 52, 51, 37, 49, 50]. TAP has recently been replaced by the *ntor* protocol by Goldberg, Stebila and Ustaoglu [52]. The *ntor* protocol is in turn derived from a protocol by Øverlier and Syverson [48].

The protocol ntor [52] as defined in Figure 3.4, is a 1W-AKE protocol between two parties P (client) and Q (server), where client P authenticates server Q. Let (pk_Q, sk_Q) be the static key pair for Q. We assume that P holds Q's certificate (Q, pk_Q). P initiates an ntor session by calling the *Initiate* function and sending the output message m_P to Q. Upon receiving a message m'_P, server Q calls the *Respond* function and sends the output message m_Q to P. Party P then calls the *ComputeKey* function with parameters from the received message m'_Q, and completes the ntor protocol. We assume a unique mapping between the session ids Ψ_P of the *cid* in Π_{OR}.

3.10 Systems Aspects and Discussion

Communication overhead. Communication overhead for BACKREF is minimal: every circuit creation, circuit extension, and stream request carries a 32 byte BLS signature and additional 4 byte timestamp.

Computation overhead. In a system with BACKREF, every node has to verify a signature and generate another. Using the pairing-based cryptography (PBC) library, a BLS signature generation takes less than $1ms$ while a verification requires nearly $3ms$ for 128-bit security on a commodity PC with an Intel i5 quad-core processor with 3.3 GHz and 8 GB RAM. Signing and verification time (and correspondingly system load) can be further reduced using faster signature schemes (*e.g.,* [60]).

Log storage. BACKREF requires nodes to maintain logs of cryptographic information for potential use by law enforcement. These logs are not innocuous, and the implications of publicly disclosing a record need to be considered. The specificity of the logs should be carefully designed to balance minimal disclosure of side-information (such as specific timings) while allowing flows to be uniquely identified. It must also be possible to reconstruct the logged data from the types

Listing 3.4 The ntor protocol

Initiate(pk_Q, Q):

1. Generate an ephemeral key pair $(x, X \leftarrow g^x)$.
2. Set session id $\Psi_P \leftarrow H_{st}(X)$.
3. Update $st(\Psi_P) \leftarrow (\mathsf{ntor}, Q, x, X)$.
4. Set $m_P \leftarrow (\mathsf{ntor}, Q, X)$.
5. Output m_P.

Respond(pk_Q, sk_Q, X):

1. Verify that $X \in G^*$.
2. Generate an ephemeral key pair $(y, Y \leftarrow g^y)$.
3. Set session id $\Psi_Q \leftarrow H_{st}(Y)$.
4. Compute $(k', k) \leftarrow H(X^y, X^{sk_Q}, Q, X, Y, \mathsf{ntor})$.
5. Compute $t_Q \leftarrow H_{mac}(k', Q, Y, X, \mathsf{ntor}, \mathsf{server})$.
6. Set $m_Q \leftarrow (\mathsf{ntor}, Y, t_Q)$.
7. Set $out \leftarrow (k, \star, X, Y, pk_Q)$, where $\star$ is the anonymous party symbol.
8. Delete y and output m_Q.

ComputeKey(pk_Q, Ψ_P, t_Q, Y):

1. Retrieve Q, x, X from $st(\Psi_P)$ if it exists.
2. Verify that $Y \in G^*$.
3. Compute $(k', k) \leftarrow H(Y^x, pk_Q^x, Q, X, Y, \mathsf{ntor})$.
4. Verify $t_Q = H_{mac}(k', Q, Y, X, \mathsf{ntor}, \mathsf{server})$.
5. Delete $st(\Psi_P)$ and output k.

If any verification fails, the party erases all session-specific information and aborts the session.

of information available to law enforcement. The simplest entry would contain the destination IP, source (exit node) IP, a coarse timestamp, as well as the signature. Logs should be maintained for a pre-defined period and then erased.

No single party can hold the logs without entrusting this entity with the anonymity of all users. The OR nodes can retain the logs themselves. This, however, would require law enforcement to acquire the logs from every such node and consequently involve the nodes in the investigation—a scenario that may not be desirable. Furthermore, traceability exposes nodes of all types, not just exit nodes, to investigation. We are aware of a number of entities who deliberately run middle nodes in Tor to avoid this exposure. An alternative is to publish encrypted logs, where a distributed set of trustees share a decryption key and act as a liaison to law enforcement, while holding each other accountable by refusing to decrypt logs of users who have not violated the traceability policy. Such an entity acts in a similar fashion to the group manager schemes based on group signatures [27].

Non-cooperating nodes. Given the geographic diversity of the ACNs, it is always possible that some proxy nodes will cooperate with the BackRef mechanism, while others will not. The repudiation property of BackRef ensures that a cooperating node can always at least correctly shift liability to a non-cooperating node. Moreover, such a cooperating node may also *reactively* decide to block any future communication from the non-cooperating node as a policy.

Venturing the last mile. In the scenarios where full traceability is required, we need a mechanism for solving the last mile problem addressed in the previous sections. BackRef does not introduce any PKI for the users, therefore our protocol has to rely on some trust mechanism to prove the linkability between the IP address of the user and the entry node pseudonym. For this purpose, we consider an ISP with a packet attestation mechanism [57] to be a proper solution that adds a small overhead for the existing ISP infrastructure and at the same time does not harm any of the properties provided by the ACN. In some countries there is an obligation for the ISPs to retain data that identify the user. In other countries the ISPs are not obligated by law, but it is nevertheless common practice. The protocol is designed in a way that the ISP has to attest only to the *ClientKeyExchange* message (this message is a part of the TLS establishing procedure, and also is public and not encrypted message) which is used to establish the initial TLS communication. This message does not reveal any sensitive information related to the identity of the user. By its design, we reuse this message as a pseudonym for the entry OR node.

3.11 Conclusion

In this chapter we presented BackRef, an accountability mechanism for ACNs that provides practical repudiation for the proxy nodes, allowing selected outbound traffic flows to be traced back to the predecessor node. It also provides a full traceability option when all intermediate nodes are cooperating. While traceability mechanisms have been proposed in the past, BackRef is the first that is both compatible with low-latency, interactive applications (such as anonymous web browsing) and does not require group managers or credential issuers. BackRef is provably secure, requires little overhead, and can be adapted to a wide range of anonymity systems. We also analyzed some important systems issues (namely, white-listing, log storage, non-cooperating nodes, and the last mile problem) with any reactively accountable ACN, and presented plausible options towards deploying BackRef in practice.

4
Oblivion

Mitigating Privacy Leaks by Controlling the Discoverability of Online Information

4.1 Motivation

Protecting privacy on the Internet remains a widely unsolved challenge for users, providers, and legislators alike. Users tend to reveal personal information without considering the widespread, easy accessibility, potential linkage and permanent nature of online data. Once the data is disclosed or leaked, the widespread is inevitable. The implications reported in the press range from public embarrassment and loss of prospective opportunities to safety issues.

Legislators have responded by tightening privacy regulations. The European Court of Justice recently ruled in Google Spain v. Mario Costeja González [67] that EU citizens have a fundamental *right to be forgotten* for digital content on the Internet, in the sense that indexing systems such as Google (or other search engines, as well as systems that make data easily discoverable, such as Facebook and Twitter) must offer users technical means to request removal of links in search results that point to sources containing their personal information and violating their data protection rights.[1] While a comprehensive expiration mechanism for digital data has often been postulated by privacy advocates in the past, this court decision, for the first time, imposes a legal constraint for indexing systems that operate in the EU to develop and deploy suitable enforcement techniques. As of now, the solution deployed by leading search engines, such as Google, Microsoft and Yahoo, consists of a simple web form that requires a user to manually identify all relevant links herself upfront and to insert them into the web form, followed by a manual evaluation by the search engine's employees to assess whether the author of the request is eligible and the request itself is lawful, *i.e.,* the data subject's right to privacy overrides the interests of the indexing operator and the freedom of speech and information.

According to the Google transparency report [68], the number of removal requests that have been submitted to Google since the court decision in May 2014 has already exceeded 700K and the number of URLs that Google has evaluated for removal are more than 2.7 millions. Clearly, in order to enable efficient enforcement, it is essential to develop techniques that at least partly automate this process and are scalable to Internet size, while being censorship-resistant by ensuring that malicious users cannot effectively blacklist links to Internet sources that do not affect them.

[1] In the court's case, the plaintiff requested the removal of the link to a 12-year old news article that listed his real-estate auction connected with social security debts from the Google search results about him. The court ruled that the indexing by a search engine of the plaintiff's personal data is "prejudicial to him and his fundamental rights to the protection of those data and to privacy — which encompass the *right to be forgotten* — [and overrides] the legitimate interests of the operator of the search engine and the general interest in freedom of information."

4.2 Contribution

We propose a universal framework, called Oblivion, providing the foundation to support the enforcement of the *right to be forgotten* in a scalable and automated manner. Technically, Oblivion provides means for a user to prove her eligibility[2] to request the removal of a link from search results based on trusted third party-issued digital credentials, such as her passport or electronic ID card. Oblivion then leverages the trust imposed by these credentials to generate eligible removal requests. More specifically, the officially-generated signatures contained in such credentials comprise personally-identifiable information of the card owner, such as her signed passport picture, address, *etc.* These so-called signed *attributes* are subsequently automatically compared with publicly available data whose removal should be requested, in order to determine if a source indeed contains information about a given entity. In Oblivion, we use state-of-the-art natural language processing (NLP) and image recognition techniques, in order to cover textual and visually identifiable information about a user, respectively. Further modalities can be seamlessly integrated into Oblivion. These techniques in particular automate the task for a user to determine if she is actually affected by an online source in the first place. The outcome of these comparisons, based on the signed attributes, is then used to provide proof to the indexing system that a user is eligibly affected by a source. To avoid creating further privacy concerns, Oblivion lets the user prove her eligibility to request data removal without disclosing any further personal information beyond what is already available at the link. This approach applies to a variety of different indexing systems, and in particular goes beyond the concept of search engines that we refer to throughout this work for reasons of concreteness. Moreover, Oblivion exploits the homomorphic properties of RSA [69] in order to verify the eligibility of an arbitrarily large set of user credentials using only a single exponentiation, and is thus capable of handling 278 requests per second on a standard notebook (2.5 GHz dual core and 8 GB RAM). We consider this suitable for large-scale deployment.

[2]With our framework we allow for the automation of the eligibility proof of the user. Eligibility in our framework describes the user being personally affected by an online source, or in legal terms being the *data subject*. The *right to be forgotten* additionally requires that the user's data protection rights override the legitimate interests of the search engine operator and the freedom of information. This assessment of the *lawfulness* of the request is a purely legal task, which is in the domain of courts. Hence the technical assessment of lawfulness is out of scope for our framework. If courts and regulators agree on guidelines for this assessment, Oblivion could be extended to a partly automated assessment of these guidelines in future work.

4.3 Related Work

The most common way to prevent *web robots* (or *web crawlers*) [70] from indexing web content is *the Robots Exclusion Protocol* (*a.k.a.* robots.txt protocol) [71], a standard for controlling how web pages are indexed. Basically, robots.txt is a simple text file that allows site owners to specify and define whether and how indexing services access their web sites. The use of this protocol for privacy enforcement is limited, since the file that defines the protocol can only be placed and modified by the administrator of the web site. The individual whose personal data is being published is hardly capable of contacting and persuading all administrators of these sources to remove the data or modify the robots.txt file. There are many attempts to approach this privacy enforcement problem in an orthogonal fashion, by adding an expiration date to information at the time of its first dissemination [72, 73, 74, 75, 76, 77]. The basic idea is to encrypt images and make the corresponding decryption key unavailable after a certain period of time. This requires the decryption key to be stored on a trusted server, which takes care of deleting the key after the expiration date has been reached. Although some of the approaches utilize CAPTCHAs to prevent crawling the images easily, there is no fundamental protection against archiving images and corresponding keys while they are still openly available, even though first successes using trusted hardware to mitigate this data duplication problem have been achieved [72]. Another approach in this direction is the concept of sticky policies [78, 79, 80, 81]. The concept was originally introduced by Mont *et al.* [78] and requires a machine-readable access policy to be bound to the data before it is disseminated. The policy then ensures that the recipient of the data acts in accordance with the policy definition. However, enforcement of such policies has to be backed by additional underlying hardware and software infrastructure. In addition to these shortcomings, a user needs to take care to augment data with expiration dates before the data is disseminated in all these approaches. Thus, these approaches are inherently unsuited to cope with data that is already openly available on the Internet or gets published by third parties. Finally, to implement the European Court of Justice's decision, Google, Microsoft and Yahoo recently launched dedicated web forms [82, 83, 84] for submitting removal requests. Users have to manually identify all relevant links and insert them into this form. Subsequently, the request is evaluated manually by the employees of the indexing system to assess first, weather the author is eligible to file that request and second, whether the link to the source needs to be deleted for a specific search. To this end, users have to additionally hand over a legible copy of an ID document. The necessity of handing over a user's full identity to use the service comes with additional privacy implications that one would like to avoid. Oblivion constitutes a technical follow-up to this solution, with a dedicated focus on censorship-resistance, while additionally avoiding the detrimental effect of having to disseminate further personal information.

4.4 Conceptual Overview of Oblivion

In this work, we propose a framework laying the foundation for a privacy-preserving automation of the *right to be forgotten* in a scalable manner. The basic idea is that users automatically identify online sources that contain their personal data and can automatically request its removal from indexing systems, if it violates their data protection rights. Upon receiving the request, we enable the indexing service to automatically verify if the author of the request is provably affected by the source in question. Our framework is sufficiently generic to incorporate any type of data, such as text, pictures, voice, and video. For brevity reasons, in this work, we mainly focus on two data types: text and pictures.

4.4.1 Motivating Scenario and System Model

We start with a motivating scenario to explain the required functionality of the framework and the different parties involved. We assume that a user, Alice, discovers that an indexing service, say Google, returns certain query requests with links pointing to a document that contains her personal information and violates her privacy. In the next step, Alice contacts an Ownership Certification Party (OCP) in order to receive validation that this source indeed contains her personal information. Such an OCP could be a third party or the Google helpdesk. Along with the relevant links, she hands over publicly verifiable ID documents such as driver's license, passport, or national ID card to the OCP. If the provided documents and the content of the article in question indeed match (which will be automatically checked by Oblivion), the OCP hands back a corresponding certificate. Alice then contacts Google to request removal of these links, providing an additional explanation, and proves her eligibility to do so based on the certificate of the OCP. Upon receiving this information, Google checks if the considered document is indeed indexed by Google, and if the OCP certificate is valid for this specific document and user. In this case, the requested article will be removed from the indexing system.

Based on this use case scenario, we consider the following entities in our proposed framework designed for automating the process of handling removal requests.

User: An authorized user who issues the request to remove her personal data.

Indexing system: This system is capable of removing links to sources containing a user's personal data from its indexing system, based on a removal request of the user.

Ownership Certification Party (OCP): It is responsible for verifying if the

user is the eligible data subject of the source under consideration.[3]

Certification Authority (CA): It issues publicly verifiable credentials to the users.

4.4.2 Threat Model and Security Objectives

We assume that all entities in the system fully trust the CA. However, a CA does not need to be online because the issuance of credentials to the users takes place out of band, typically for a longer period of time, say a couple of years.

Unlike the CA, the OCP is an entity that is not fully trusted from the user's perspective because it can try to learn the user's keying material and additional user credentials not required for the ownership verification; moreover, it might want to forge removal requests. The OCP is the only entity that is not part of the traditional system. The OCP can be run by the organization (*e.g.,* Google) that manages the indexing system, or it can be a third-party service. The OCP is assumed to be online during the execution of a request.

The indexing system is an entity inherently present in the traditional system. The indexing system and the OCP mutually trust each other; in practice, this is often trivially the case since the OCP and the indexing system are often managed by the same organization. If the OCP is an independent third party, this trust would typically be established via the CA using appropriate certificates.

We assume that users protect their private keys or at least, if their private keys are lost or stolen, a key revocation mechanism is installed and the user generates new keys. During the ownership verification, we do not assume any interaction between the users and the OCP. A user can present the OCP-signed proof to remove links to the data from multiple indexing systems, such as Google and Yahoo. We also consider an external adversary that could harm credibility of the user through replay attacks with the intention to make the service unavailable. For providing confidentiality over the communication network, we assume the presence of secure channels (such as SSL/TLS [85]) between a user, the OCP, and the indexing system.

Based on these assumptions, we intend to achieve the following security objectives:

Minimal Disclosure: An indexing system should not learn anything beyond what is required for eligibility checking and assessment of lawfulness. The court decision ruled that the right has to be judged on a case-by-case decision. Hereby, the right of the individual has to be balanced with the public right of information. Our system handles removal requests that prove eligibility

[3]Ownership in this context should not be confused with the legal term. Legally, the OCP can only assess and certify the individual's eligibility since, at least in EU context, legal ownership is not applicable to the right to be forgotten.

but do not reveal any further information beyond what can be found in the online source in question.[4]

Request Unforgeability: The system should be designed such that an indexing system can only verify user requests without any possibility of forging existing or generating new requests on behalf of the user.

Censorship-Resistance: The system should prevent censorship in the sense that only requests from provably affected users should be taken into account.

In addition to ensuring these security properties, the system should satisfy the following system properties in order to be suitable for large-scale deployment. It should be *scalable* in order to be able to process a large amount of queries simultaneously, while at the same time ensuring a thorough treatment of each individual query. It should *blend seamlessly into existing infrastructures*, to enable adoption by current indexing systems and certification authorities; moreover, the solution should be conceptually independent of the device and the operating system used. Finally, it should be *easy to understand and use* even for the general public.

4.4.3 Key Ideas of the Protocol

Oblivion is built on top of already available infrastructure (as explained in Section 4.4.1) that includes users, an indexing system and a CA. For the automatic verification of ownership, we introduce only a single new entity, the OCP, thus making our framework deployable in practice. In the framework, we distinguish three main phases: registration, ownership claim, and reporting phases. Figure 4.1 presents the overall architecture for achieving the goals defined in Section 4.4.2.

Registration Phase. During the registration phase, each user registers with the CA as shown in Figure 4.1. For the registration, a user presents (in Step 1) her attributes (along with evidence) and her verification key. The verification key should, for privacy reasons, be generated by the user herself before contacting the CA, but the generation of the key is orthogonal to our framework. The CA checks the validity of the attributes presented, certifies them and returns (in Step 2) a list of signed attributes, where each signed attribute is bound with the user's verification key. Typical examples of attributes are the date of birth, name, or a user's profile picture.

[4]Although Oblivion provides for minimal disclosure, the indexing system might request additional information, such as an author's name, for liability reasons in a real-world deployment of Oblivion. Moreover, the assessment of lawfulness could in some cases also require additional personal information.

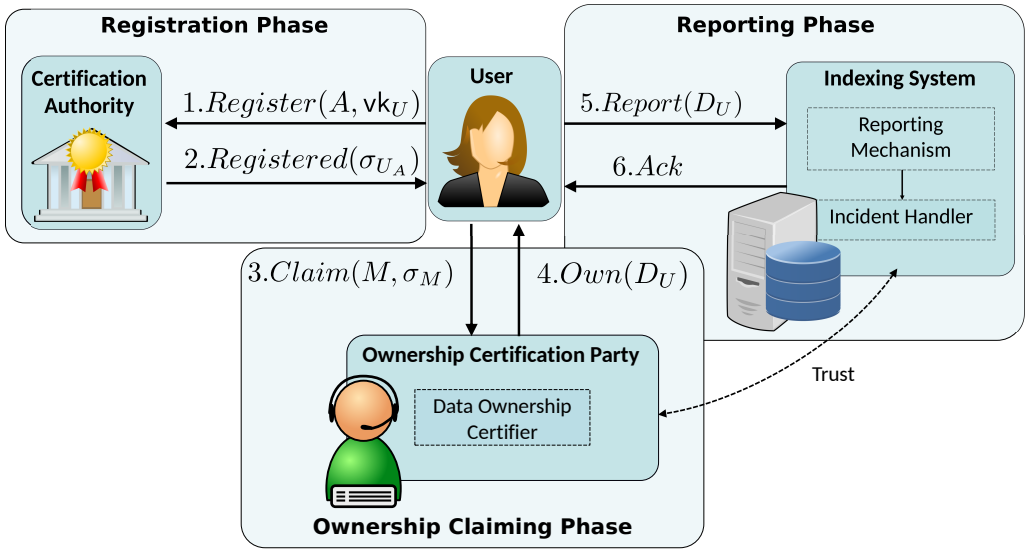


Figure 4.1: Conceptual overview of OBLIVION.

Ownership Claim Phase. Once a registered user finds leakage of her personal data through the indexing system, she can contact the OCP claiming eligibility (in Step 3). This is the core phase in which the OCP expects justification of why the given piece of data affects the user. To make such a justification, the user can put tags on the given data that consist of her attributes which were signed by the CA. In order to improve usability, we automate the tagging and verification. One trivial automation method is to simply check if any user attribute appears anywhere in the article; if this happens, the matched item could be tagged with that attribute. The name attribute, say Alice, could be matched in this way.

The exact matching can semi-automate the tagging process but it cannot work in general because it may not return the correct results for all user attributes. Let us consider a user attribute in the form of a tuple: ⟨Nationality, German⟩ (as explained in Section 4.5.1). In order to match this attribute, the OCP has to check if the user attribute or its synonym has appeared in the article. This includes semantically linkable formulations, such as *being a citizen of Germany* and *having German nationality*.

Letting the user manually deploy this solution, *i.e.,* forcing the user to find synonyms of each possible word in the article, is an exhaustive task. Therefore, we employ an NLP-based technique — the named entity recognizer (NER) [86] in our case — for efficiently collecting all possible candidates in the article. The NER detects and classifies the data into various categories, such as person, organization, location, date and time, and it thus helps to identify if a user has attributes belonging to the category identified by the NER. If yes, we can perform exact matching or run a synonym checker [87] on identified categories. Articles containing a user's picture are tagged in a corresponding manner.

After the attributes are matched, the user has to generate a proof by preparing a message that contains a list of signed attributes that are required for the verification, the tagged article and her verification key. The user signs this message and sends it to the OCP (in Step 3) as an eligibility claim.

The OCP first verifies the message signature and the signed attributes used in the tagging. If the claim relates to text attributes, the OCP runs an entity disambiguator to identify whether the article is about the user. If the claim includes a picture, the OCP runs a corresponding face recognition algorithm. Upon successful evaluations of all steps, the OCP presents to the user an ownership token (in Step 4).

Reporting Phase. After receiving the ownership token from the OCP, the user sends a request for removal to the indexing system (in Step 5). The indexing system automatically validates the ownership token and then assesses whether to remove the links pointing to the user's personal information from its system. Finally, it sends (in Step 6) an acknowledgment to the user, which could be a *success* or *failure* message.

4.5 Realization Details of Oblivion

In this section, we provide details of each phase of our framework and explain the communication protocol to show interaction between different components. An indexing system and a user are denoted with IS and U, respectively. The communication protocol steps, described in this section, correspond to the flow illustrated in Figure 4.1. After that, we provide details on how to securely and efficiently realize the proposed protocols using cryptographic primitives.

4.5.1 Registration Phase

As we can see in the communication protocol, a user sends (in Step 1) her attributes, $A = \{a_1, a_2, \ldots, a_n\}$, which characterize her, with supporting proofs and the verification key vk_U to the CA. Each user attribute $a_i \in A$ is a name and value key pair $\langle \mathsf{NAME}, \mathsf{VALUE} \rangle$, representing name of the attribute and value specific to each user, respectively. For instance, an attribute name could be *National*, and if say, a user is national of Germany, then the value will be *German*. Some general user attribute names include, but are not limited to, *Full Name*, *Date of Birth*, *Place of Birth*, *Current Residence*, and *ID Picture*.

Upon a successful verification of the provided data, the CA issues a list of signed attributes $\sigma_{U_A} = \{\sigma_{U_{a_1}}, \sigma_{U_{a_2}}, \ldots, \sigma_{U_{a_n}}\}$ and sends it back to the user (in Step 2). Our attribute signing scheme binds every user's attribute with her verification key. Note that one of the attributes a_i is a profile picture that uniquely identifies the user.

Steps 1 and 2 constitute the registration phase that takes place securely and out of the band. The concept of digital signature together with user attributes (signed by the government) is already present in some EU countries [88, 89, 90].

4.5.2 Ownership Claim Phase

In order to make an ownership claim to the OCP, we consider a user client, say a browser plugin. The plugin sends the claim to the OCP and receives an ownership token from the OCP in the case the claim can be verified, cf. Figure 4.1. In order to do so, the first step is that the user client has to formulate the claim, then it has to identify personal information and finally the actual removal request has to be generated. In the next step, the OCP has to verify the request. This is done by first verifying the authenticity of the request and second verifying the relationship to the data. The latter verification depends on the type of data, *e.g.,* face recognition can be used for pictures. The last step is to generate the ownership token that is then transferred from the OCP to the user. In the following, we present the details of all these tasks.

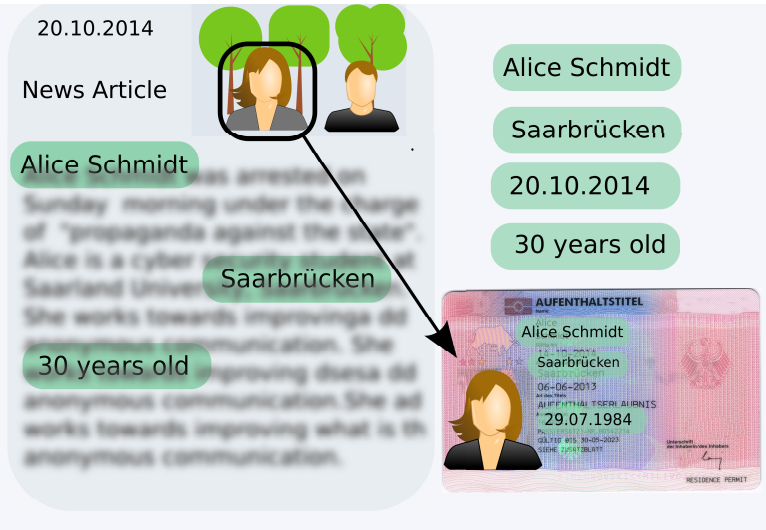


Figure 4.2: An article illustrating personal information of Alice Schmidt who has an ID card with digital credentials issued by the German government.

Identifying Personal Information. For identifying user's personal information in an article (as illustrated in Figure 4.2), a user client may run the NER algorithm locally (assuming it is delivered as a part of the user client) to extract all possible candidates. The NER algorithm could also be run as a third-party service (*e.g.,* a web service), called by the user client. After running the NER algorithm, a user client picks each of the candidates and matches them with the user attributes (see Figure 4.2).

If the match is not successful, a user client runs a synonym checker. If both words are synonyms then they are considered matched; otherwise, the next candidate is picked from the queue for the comparison. The synonym checker could be delivered as a part of the user client. To make the user client lightweight, we can assume a third party service (*e.g.,* a web service). In either case, the synonym checker should be very specific to the attributes issued by the CA.[5]

Face Detection. Besides the textual description, an article could also contain a user's picture, either as a solo or a group picture. Like textual attributes, the user client can run the face detection algorithm to automatically detect the user's face. On successful detection, a user client can automatically include the CA-signed user picture in the removal request, which is explained next.

Generating Removal Request After identifying personal information, a user client prepares a removal request. During the preparation, it chooses all signed attributes required for the ownership claim. Next, it packs them as $P_{\sigma_{U_{A^*}}}$ so that the OCP can verify the signed user attributes using a single exponentiation operation using the CA verification key. This would also require a user client to include in the message a subset of her attributes A^* corresponding to the packed ones, *i.e.,* $P_{\sigma_{U_{A^*}}}$. Since a user client signs the message using the user's signing key, the user's verification key vk_U is also included in the message to let the OCP verify the message. For preventing replay attacks, a timestamp TS is also included in the message. The user client sends to the OCP (in Step 3) the message $M = (TS, \mathsf{vk}_U, A^*, P_{\sigma_{U_{A^*}}}, D)$ along with the signature σ_M.

Verifying Removal Request. Upon receiving a removal request, an OCP verifies it before issuing any ownership token. As a first step, the signature σ_M over the message M is verified. Next, the OCP checks the timestamp and verifies the packed version of the user attributes signed by the CA. Then, the OCP checks if all tagged attributes are valid. This step comprises the exact matching and/or synonym checking.

Face Recognition. Optionally, the face recognition algorithm could be run provided there is a user picture in the article. As we explained earlier in this section, faces are pre-identified by the user client, in order to ease the job of the OCP. The OCP compares the user-tagged face with one provided as a signed user attribute in the request (see Figure 4.2). If the face recognition algorithm

[5]For instance, the user's date of birth might appear differently in an article, *i.e.,* in the form of her age as shown in Figure 4.2. If this happens, the age could be compared with the difference of the user's date of birth and publication date of the article, if present. As we can see in the example, *30 years old* will be compared with *20.10.2014 - 29.07.1984*. Further tests for checking syntactic equivalence are conceivable, but are postponed to future work.

discovers similarity with a certain confidence, the user's picture in the article is considered matched with her profile picture.

Entity Disambiguation. When the given article contains text, the OCP can execute the disambiguation algorithm (*e.g.,* AIDA [91]) for ensuring the eligibility goal, *i.e.,* checking whether the article is about the user. The outcome of this algorithm is the relation between the user attributes, her name in particular, and the context of the text. The outcome, say satisfying the predefined threshold value, would help the OCP to mark the user as being affected by the data in the article. Figure 4.2 illustrates an example article about Alice Schmidt.

Issuing Ownership Token. On successful evaluations of all the steps performed by the OCP, the user is issued an ownership token. This is accomplished by the OCP by sending (in Step 4) an ownership token D_U to the user. It is important to note that the OCP verification protocol is non-interactive.

4.5.3 Reporting Phase

Once the user receives the ownership token, she can report to the indexing system. In this phase, a user reports by sending (in Step 5) the ownership token D_U (corresponding to D) to the indexing system. The indexing system verifies the token, fires the incident and sends (in Step 6) an acknowledgment *Ack* to the user. If the OCP is a third-party service, the ownership token is signed by the OCP and could be sent to multiple indexing systems simultaneously.

4.6 Performance Analysis

In this section, we provide implementation details for all components that we newly developed for Oblivion and name libraries that this implementation relies on. We subsequently evaluate the performance overhead of this implementation for each involved component (CA, user client, and OCP).

4.6.1 Implementation Details and Evaluation Parameters

Components of the Implementation. The implementation prototype is written in Java. To reflect the different involved participants, the implementation consists of three components: a module for the CA (CA-module), a module for the OCP (OCP-module), and a module for the user client (user-module). For the sake of simplicity, the prototypical implementation assumes that the OCP and the indexing system are managed by the same organization; this avoids an additional trust level between these institutions and allows us to concentrate

on the performance measurements. The size of each of these modules (without included libraries; see below) is below 5KB.

Libraries Used. Our prototypical implementation relies on several existing open source libraries. First, we include the Stanford NER library [92] for identifying personal information in the textual article. The NER library is of size 3.2 MB and the NER classifier, for covering seven distinct classes of data, requires 16.6 MB. Second, we rely on OpenCV (Open Source Computer Vision Library), an open source computer vision and machine learning library [93], for face detection and recognition. Finally, we include the AIDA (Accurate Online Disambiguation of Named Entities) framework [91] to achieve ownership disambiguation. In our experiments, we used the AIDA framework itself and its corresponding web service, which works with entities registered in the DBpedia [94] or YAGO [95] knowledge base.

Evaluation Parameters. We have evaluated the performance of the implementation on a dataset of 150 news articles that we randomly crawled from the international news agency Reuters[6], using the Java-based web crawler *crawler4j* [96]. These articles cover different topics and range from 1K to about 10K words; the average length is 1.9K words per article. The actual experiments were run on a standard notebook with 2.5 GHz dual-core processor and 8 GB RAM. The experimental results described below constitute the average over 100 independent executions. Network latency was not considered in the experiments.

4.6.2 Evaluating the CA-Module

Evaluating the performance of the CA-module consists of measuring the overhead of attribute certification.

Attribute Certification. Figure 4.3 illustrates the computational overhead for certifying user attributes. In our experiment, we generated up to 50 attributes and considered CA's signing keys of varying size, ranging from 512 to 4096 bits. As we expected, certification time grows linearly in the number of attributes. For the most complex cases under consideration — the CA signing 50 attributes, and thus far more than what a user would typically maintain, using a signing key of size 4096 bits — the attribute certification took 7.5 seconds. For smaller numbers of attributes, or for all smaller key sizes, this certification takes less than a second. Since attributes are typically certified only once per user, this computational overhead should be acceptable as a one-time upfront effort.

[6] `http://www.reuters.com/`

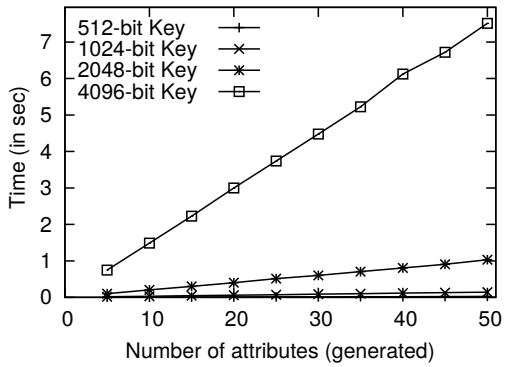
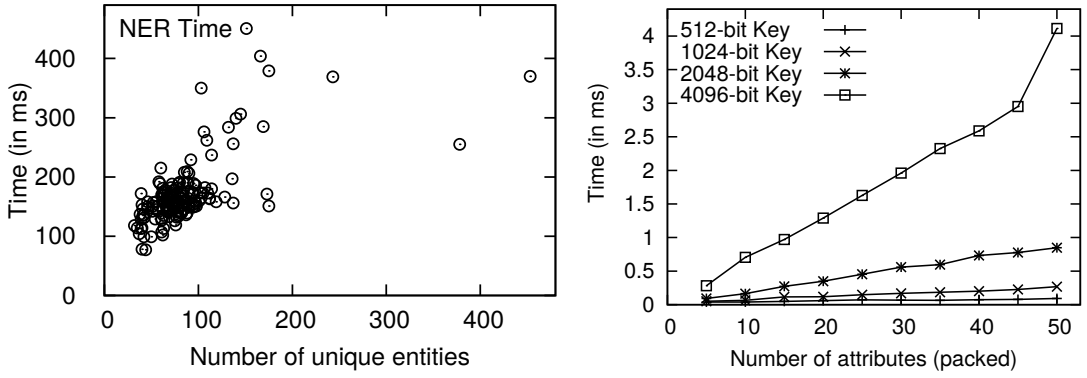


Figure 4.3: Evaluation of the CA-module: Performance overhead for certifying user attributes.

4.6.3 Evaluating the User-Module

Evaluating the user-module is performed in two steps: identifying suitable attributes in the given sample texts, and pre-processing these attributes for the subsequent ownership-proof phase.



(a) NER overhead (b) Packing overhead

Figure 4.4: Evaluation of the user-module: Performance overhead of (a) identifying personal information and (b) for packing user attributes.

Identifying Attributes. As explained in Section 4.4.3, the user-module preprocesses the article using NER techniques and appropriately selects all entities that are necessary for the identification process. We evaluate the performance of the user-module on the aforementioned 150 news articles from Reuters, and measure the time required to identify and extract all entities. The results are depicted in Figure 4.4(a). The performance overhead varies from 77 to 814 millisecond (ms), with an average of 174 ms per article. The number of unique

entities in the articles ranges from 43 to 590, where the average number of unique entities per article is 135.

Attribute Packing. After identifying all personal attributes in a given news article, the user-module pre-processes a set of signed attributes as required for the ownership proof. This pre-processing in particular reduces the number of exponentiations that are required to verify the attributes for the OCP, and thereby avoids a potential bottleneck. In the performance measurement, we again considered up to 50 attributes and varying key sizes. As shown in Figure 4.4(b), the time for this pre-processing increases linearly in the number of attributes, with an additional overhead for larger key sizes. For the maximum of 50 attributes, the pre-processing only took between 0.1 ms (for a 512-bit key) and 4.1 ms (for a 4096-bit key).

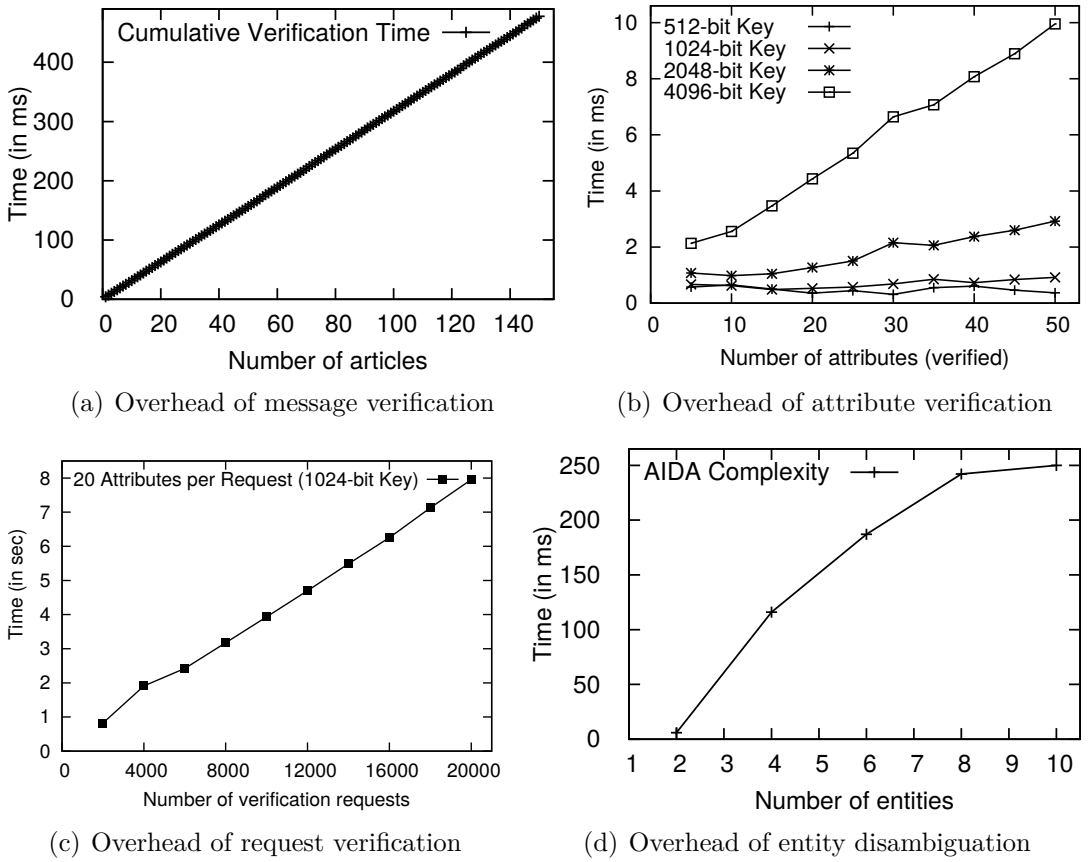
Message Signing. The user client signs the message using her signing key. For this experiment, we considered the aforementioned 150 news articles. Consider the overhead of signing a message with a signing key of size 1024 bits. Depending on the size of the article, the signing took between 2.8 and 3.8 ms, with an average of 2.9 ms per article.

4.6.4 Evaluating the OCP-Module

We split the performance evaluation of the OCP-module into two parts: First, we evaluate the time required to verify the validity of requests for varying parameters: for varying numbers of articles, for varying number of attributes, and for varying verification requests. Second, we evaluate the time required to decide whether the request is legitimate, *i.e.,* whether the document under consideration affects the user's data, either by means of entity disambiguation or face recognition.

Validating the User Request. Upon receiving a signed message from a user, the OCP verifies the validity of the signature using the user's verification key. This verification time (with a 1024-bit key) ranges from 2.9 to 4.3 ms with an average of 3.2 ms per article. Figure 4.5(a) illustrates the cumulative verification time to verify up to 150 articles. It grows linearly, so verifying message validity for 150 articles takes the OCP less than 0.72 seconds.

Similarly, Figure 4.5(b) displays the time required to verify a certain number of signed user attributes. Recall that the user sends a packed version of her signed attributes to ease the verification task of the OCP. Still, the OCP needs to calculate the hash of each individual attribute and multiply all hashes together before being able to verify the signature based on the packed version. Verifying 50 user attributes takes 0.37 ms (for a 512-bit key) and 10 ms (for a 4096-bit key),



(a) Overhead of message verification

(b) Overhead of attribute verification

(c) Overhead of request verification

(d) Overhead of entity disambiguation

Figure 4.5: Evaluation of the OCP-module: Performance overhead of (a) verifying the messages, (b) verifying user attributes signed by the CA, (c) verifying user requests and (d) running entity disambiguation.

respectively. For l attributes, the packed version is at least $l - 2$ exponentiations faster than verifying each attribute individually.

Finally, Figure 4.5(c) shows the performance overhead for verifying a certain number of user requests. In our experimentation, we assumed that every request requires the verification of 20 attributes, each one signed with a key of size 1024 bits. To measure the performance overhead, we gradually increased the number of user requests from 2,000 to 20,000 and observed an (essentially linearly-growing) overhead from 0.824 to 7.96 seconds. Processing a single verification request with 20 attributes took less than 0.4 ms on average.

The overall computational overhead of the OCP-module is a combination of the message verification and the attribute request verification, each one incurring on average 3.2 ms and 0.4 ms, respectively. Therefore, our implementation manages to process a removal request within 3.6 ms. In summary, it allows the OCP to handle 278 requests per second (using the standard laptop that we based these experiments on).

Eligibility of the User Request. Identifying whether the requested article indeed contains personal data of the requesting user relies on appropriate entity disambiguation. Figure 4.5(d) illustrates the performance overhead for entity disambiguation with up to 10 entities.

Recall that we require the user client to run the face detection algorithm and select the appropriate face and send it the OCP along with the standard request. The performance overhead of the face recognition algorithm depends on multiple factors such as the picture resolution and the face position in the picture. In our experiments, we have chosen pictures with well-defined frontal faces. The resolution of the pictures is up to 3,072 x 4,608 pixels with an average size of 4 MB. Having all these predefined conditions, the runtime of the face recognition algorithm stays in the range of 150 to 300 ms.

The overall performance overhead, comprising both entity disambiguation and image recognition, currently constitutes the bottleneck for verifying the validity of removal requests in the OCP-module. Currently, we are exploring further optimization here.

4.7 Security Analysis

The framework is supposed to achieve three security objectives: minimal disclosure, request unforgeability, and censorship-resistance, as discussed in Section 4.4.2. In the following, we show that we achieved these goals.

Minimal disclosure. Minimal disclosure in this context is the minimization of knowledge increase for the indexing system in order to verify eligibility of a request. In the case that OCP and IS are separate, this is given since the IS only receives a token from the OCP through the user. This token does not need to contain any information about the user. However, if the OCP and the IS collide, it has to receive the input to run OCP.VerifyA.

A user who wants to hide her verification key or credentials could potentially use interactive zero-knowledge proofs on top of our construction. This, however, would sacrifice efficiency and would not improve the disclosure of information. The reason is that the verification key is basically a pseudonym, *i.e.,* it is only linked to the attributes that we send. Thus, the only need to minimize is the sending of attributes. In Oblivion, we only send those attributes that are indeed necessary for proving that the user is affected, *i.e.,* that already occur in the link we report. We implement this by sending only subsets.[7]

[7]We stress again that we only show affectedness of the user. Arguing about the legal implications and whether this minimization of data is sufficient in order to apply them is beyond the scope of this (and all existing) work.

Request Unforgeability. For unforgeability we show that even the user cannot construct a message that verifies without having a signature on every single attribute. As a consequence, the user cannot show that she is affected by content concerning other users' attributes.

Theorem 5 (Request Unforgeability)**.** *If* OCP.VerifyA *returns* accept *then the packed attributes correspond to the set* A^**. More formally, every* $\mathcal{A}$ *that has access to a signing oracle* $\mathcal{S}$ *with public key* vk *can only generate* P *for subsets* A^* *of all signatures* A *requested from* $\mathcal{S}$*.*

Proof. Let A be the set of queried attributed signatures of the adversary $\mathcal{A}$ for a given execution. Assume there is a set $(B, P) \xleftarrow{\$} \mathcal{A}^{\mathcal{S}}(\mathsf{vk}_u)$ such that $\mathsf{OCP.VerifyA}(\mathsf{vk}, \mathsf{vk}_U, P, B) = \mathsf{accept}$ and $B \not\subseteq A$. So there exists $b^* \in B$ such that $b^* \notin A$. Then there also exists an adversary $\mathcal{A}^*$ that queries $A \cup B \backslash \{b^*\}$, *i.e.,* b^* is the only unqueried attributed in B. Since $\mathsf{OCP.VerifyA}(\mathsf{vk}, \mathsf{vk}_U, P, B) = \mathsf{accept}$, it follows that $P^{e_{CA}} \equiv \prod_{b \in B} \mathcal{H}(b||\mathsf{vk}_U)$ by construction. Since we queried all b except b^* in B, we can compute $\sigma := P / \prod_{b \neq b^* \in B} \mathcal{H}(b_i||\mathsf{vk}_U)^{d_{CA}}$. For this σ, we have $\sigma^{e_{CA}} = \mathcal{H}(b^*||\mathsf{vk}_U)$. However, this contradicts the Chosen Message Attack (CMA) security of the underlying signature scheme. Thus, the adversary $\mathcal{A}$ cannot exist. □

Censorship-Resistance. Finally, we have to ensure that the overall system does not enable any user to censor, *i.e.,* to successfully report data that she is not affected by. There are two possible approaches. First, we could do a reduction proof to the CMA-security of the signature scheme as done for the request unforgeability.[8] Second, we can formulate the protocol in the applied π-calculus and automatically verify the properties of interest using tool support. The outcome can then be leveraged from the protocol to the implementation by using computational soundness which links symbolic execution traces to computational execution traces. Thus, we can use tools for symbolic verification and the outcome transfers to the implementation. In what follows, we pursue the second approach since the protocol is easy to express and verify using state-of-the-art verification tools.

[8]Such a proof would look like this: Assume censorship is possible. That means there is an execution that ends with a successful report at the indexing system without the user reporting the data. Therefore, there was a D_U sent to the IS that verifies with the key of the OCP. Either the OCP signed D_U or there is a contradiction to the signature scheme's CMA-property. Consequently, the OCP signed D_U and since we assume the OCP to be trustworthy, it means that the OCP received an M, σ_M from a user and verified it. Here, either the user's signature σ_M was forged (contradicting the CMA-property of the signature scheme) or the user forged a message M that verifies (contradicting the request unforgeability proven before). It follows that the user could not have generated such a request, proving censorship-resistance.
While this argumentation sounds plausible, it does not consider every possible interleaving or repetition of executions. In contrast, tool support offers a trustworthy guarantee that we did not overlook any execution generated by these processes.

The applied π-calculus defines a way of modeling processes (`P,Q`). Thereby, the calculus gives constructs for parallel execution of processes (`P|Q`), for repetition of processes (`!P`), for communication between processes (`in(chan,msg),out(chan,x)`) and for restricted computation. The restriction is that only symbolic constructors (`let x=sig(sk,msg)`) and destructors (`let m=verify(vk,sig)`) can be used to modify terms which consist of symbols. The difference is that constructors create symbolically larger symbols, *i.e.,* in the example x will be handled as the symbol `sig(sk,msg)` whereas destructors can give reduction rules to remove or replace constructors. Finally, for symbols, there are two classes, publicly known symbols and freshly introduced symbols (`new N; P`) which are unequal to all other symbols.

For the sake of exposition, we briefly describe the process for the indexing system (IS). The system receives a message and verifies it with the corresponding key (computational soundness requires the key to be part of the signature). We then verify the first part of the signed message with the verification key of the OCP. This message must be the user's verification key and the requested data, *i.e.,* we check for equality before the IS is convinced that the signer is affected.

```
let IS = in(ch,x); let tmpkey = vkof(x);
let (c,reportData) = verifySig(tmpKey, x) in
let (sigKey, sigData) = verifySig(vkOCP, c) in
if reqData==sigData then if tmpKey==sigKey then
event affected(sigKey,reqData).
```

The end of the process is a so-called event. These events have no semantic meaning in the calculus, but can be used by the model checker to prove certain properties of the protocol. In our example, the event symbolizes the belief of the IS that the user with verification key `sigKey` is affected by the data `reqData`. The model checker answers queries such as `query ev:affected(k,d).` which formalizes that the model checker can prove that this event can be reached in the protocol execution.

Censorship-resistance can be formulated as a sequence of events that has to occur whenever the IS thinks a user is affected, *i.e.,* whenever a request is considered to affect the requesting user, the OCP has verified that this data belongs to the user that sends the request. This can be done by two queries of the form `query ev:affected(key,d) ==> ev:VerifiedOw(x,key,d).` meaning that whenever the event `affected` occurs there has to be a corresponding event verifying the ownership beforehand. Analogously, we prove that the ownership verification is preceded by the attribute verification of the CA.

The complete formalization in the applied π-calculus can be found online at the project website[9]. The protocol verification takes 8 ms.

[9] `https://infsec.cs.uni-saarland.de/projects/oblivion/`

Other Security Goals. In order to prevent a replay attack, the user includes the timestamp in her request. One can argue that a replay attack is not an issue because it is a legitimate request by the authorized user. However, we consider that a replay attack could harm the credibility of the user if an adversary launches it to mount a Denial-of-Service (DoS) attack on the OCP.

4.8 Discussion

Deployability and Usability. In order to deploy our solution, OBLIVION requires a national or local government-wide CA that issues credentials to citizens. We argue that this requirement does not limit practicality of our approach because the issuance of such credentials is already part of an EU standard [90], implemented by some member states and meant to be adopted by all the EU member states [88, 89]. The European EID standards also enable the use of digital credentials for Internet communication (*e.g.,* for online shopping) [88] which also strengthens usability for OBLIVION's developers as well as end-users.

Scope of Eligibility. First, it is a hard problem to decide on the eligibility of an ownership claim if two persons have the same attributes, *e.g.,* name. OBLIVION addresses this issue by using attributes that in combination should be sufficiently unique for most people. Second, our framework cannot decide whether a piece of content is of public interest (such information falls into the category of freedom of the press) and outweighs the privacy interest of an individual. This decision is a legal assessment. This is outside of the scope of OBLIVION and subject to ongoing research about the automation of legal assessments [97].

Privacy and Availability. The OCP could be a third-party service or managed by the search engine provider. From a privacy point of view, the latter setup may reveal personal information about citizens. However, we argue that a search engine provider does not learn more than what is already available in the article. This is because OBLIVION follows a principle of least privilege, where only those particular attributes that are present in the article are sent to the OCP. The collection of information and verification makes the OCP a key component of OBLIVION. The availability of the OCP becomes essential in the long-run success of OBLIVION. Therefore, to prevent a single point of failure, we can consider deploying multiple instances of the OCP.

Robustness. OBLIVION relies on NLP and image recognition techniques. The NLP technique we use in our framework is simple and sufficiently robust in practice. Concerning robustness of the image recognition technique, recent research has shown that automated face recognition is almost comparable to human face

recognition accuracy [98]. Therefore, when the removal request includes a picture that uniquely identifies the user with a certain confidence (part of the deployed policy), our framework can easily approve the removal request.

4.9 Conclusion

In this chapter, we have introduced a universal framework, providing the foundation to support the enforcement of the *right to be forgotten* in a scalable and automated manner both for users and indexing systems. The framework enables a user to automatically identify personal information in a given article and the indexing system to automatically verify the user's eligibility. The framework also achieves censorship-resistance, *i.e.,* users cannot blacklist a piece of data unless it affects them personally. This is accomplished using the government-issued digital credentials as well as applying the entity disambiguator technique. We have conducted comprehensive evaluations of Oblivion on existing articles, showing that the framework incurs only minimal overhead and is capable of handling 278 removal requests per second on a standard notebook (2.5 GHz dual core). In these evaluations, we have observed that the remaining performance bottleneck on the OCP is caused by the entity disambiguator (*i.e.,* AIDA) and the face recognition (*i.e.,* OpenCV) algorithms. We believe that optimized versions of both could help in significantly improving the performance.

5

Who Controls the Internet?

Analyzing Global Threats using Property Graph Traversals

5.1 Motivation

About half of the world population is using the Internet every day to communicate with friends, read newspapers, and carry out financial transactions. These services rely on core operations such as IP routing, domain name resolution, and email transfers, which are carried out by organizations ranging from universities and governmental agencies to private-sector organizations. Such organizations thus have extensive power, which, if misused, can result in global-scale security violations. Service providers can perform various attacks such as advertising false BGP paths to sensitive targets through their network [9, 10] and injecting HTTP responses into TCP connections [11]. Even more severe security violations can be performed when providers cooperate. Recent events have shown that cooperation between providers and state authorities resulted in global-scale security incidents such as mass surveillance, *e.g.,* the PRISM program [99], and distributed denial-of-service (DDoS) attacks, *e.g.,* the Great Cannon attack [12, 100].

Service providers can also be victims of attacks. The Internet is often considered as a model of resilience due to its distributed and decentralized design. While this applies in cases of random node failures, it does not guarantee survivability of the network with targeted attacks [101, 102]. For example, attackers can focus their efforts against a few, carefully selected providers to disrupt network operations at large scale. These types of attacks have already been observed against root name servers, the servers at the top of the DNS hierarchy, so far they have had very limited impact. However, the skills and power of attackers are increasing, and the DNS infrastructure of Dyn.com, which serves popular websites, was struck by two DDoS attacks. While the volume of the attack was never officially confirmed, this attack caused outages in the name resolution of popular services such as Amazon, Netflix, Twitter, Reddit, and Spotify.

An increasing number of reports and studies are showing that a limited number of players have an important influence on the overall security of the Internet infrastructure. While we have a good understanding of attack techniques [11, 12], attackers [103], and victims [104], we have a rather limited capability to assess the impact of attacks against, or performed by, core service providers. In the past decades, the security of the Internet core infrastructures has been under continuous scrutiny. Many works focused on different facets, using analysis techniques such as topological analyses (*e.g.,* [105, 106, 107]) and traditional threat analysis via attack enumeration (*e.g.,* [108]). However, the contribution of these works is limited to a single core service, and, to date, the interdependencies between core services remain largely unexplored.

5.2 Contributions

In this work, we take a step forward and propose an investigation technique to assess global-scale threats. We present a model of the Internet infrastructures based on *property graphs*. Nodes of our model are servers, organizations, and autonomous systems, that are connected with edges to represent relationships. To mine data from our model, we present a combination of taint-style techniques and propagation rules, which is automatically translated in graph traversals. We assessed our approach on a model with 1.8 millions of nodes and 4.7 millions of relationships. Starting from the top 100K Alexa domains, we built our model using publicly available resources, *e.g.,* RIPE Atlas, and by acquiring relationships via Web service crawling. Finally, we mined our model to assess the impact of attacks. We present six metrics to select attacker and victim candidates. Then, we measure the impact of three different attack scenarios, which are based on the Great Cannon attack, the PRISM program, and the DDoS against Dyn.com.

Our results show that already just a few players may have an extensive power: 14 countries and 14 autonomous systems can, directly or indirectly, affect the security of about 23% of websites. Our analysis show that the United States is the country with the largest fraction of power, *i.e.,* 16% of websites, and that network operators, albeit of moderate size such as Google[1], can match in terms of affected websites the aggregate result of large countries like Russia, Germany, Japan, and China. In addition, our results show that little has been learned from past attacks. For example, 70% of JavaScript (JS) inclusion is still done over unprotected connections, *i.e.,* via HTTP URLs, which can be used to mount the Great Cannon attack. Finally, our results indicate that the DDoS attack against Dyn.com was the result of careful choice. The autonomous systems that belong to Dyn host authoritative name servers used directly and indirectly by 3% to 5% of the 100K Alexa domains.

To summarize, this work provides the following contribution:

- We present a first study on attacks based on dependencies between Internet core services;
- We present a framework to model and reason on global-scale threats;
- We present a taint-style technique based on propagation rules and property graphs to quantify the impact of security incidents;
- We assess our technique on 1.8M data items acquired from the top 100K Alexa domains.

[1]Google is also a network operator as it controls the autonomous system 15169

5.3 Background

Before presenting our framework, we describe relevant case studies and introduce our threat model.

5.3.1 Case Studies

Our study is motivated by three recent, large-scale and well-known security incidents.

The Great Cannon DDoS Attack. On March 16th, 2015, Greatfire.org, a non-profit organization monitoring Internet censorship in China, and GitHub, the hosting provider, were victim of a large DDoS attack, among the largest DDoS ever experienced by GitHub [100]. The attack was caused by malicious JavaScript code, which was injected into TCP connections crossing the Chinese network borders [12, 109]. The injected code turned Web browsers into an HTTP-based DDoS botnet by aggressively requesting resources from the targets [12, 109, 110].

The PRISM Program. On June 7th 2013, *The Guardian* documented PRISM, a National Security Agency surveillance program with direct access to Internet communications and stored information including emails, chats, and VoIP calls, from servers of popular tech companies such as Microsoft, Yahoo, Google, and Facebook [99]. While the direct involvement of popular tech providers is still unclear, in this work, we make the assumption that establishing this type of collaboration is possible and can be voluntary, or coerced by authorities by means of law and court orders.

The DDoS Attack Against Dyn.com. On October 21st 2016, the DNS infrastructure of Dyn.com was struck with two DDoS attacks. According to Dyn.com,[2] the attack caused increased DNS query latency and delayed zone propagation. As a result, Dyn.com customers, including Amazon, Netflix, Twitter, Reddit, and Spotify, experienced outages on the name resolution. Dyn.com later reported that the attack was sourced from 100,000 malicious endpoints which were part of the Mirai botnet. The attack was complex, using maliciously targeted, masked TCP and UDP traffic [111].

5.3.2 Threat Model

Our threat model is motivated by the case studies presented in the previous section. Our focus is on a large scale attacks that can target big number of individuals and

[2] `http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/`

organizations around the globe. We represent attacks as a set of three elements: an attacker, the attack goal, and the attack technique.

Attacker. Attackers can be a service provider, a group of providers, or a country. In case the attacker is the provider, we consider the attackers: domain name provider, email provider, network provider, content distribution network, and domain name owners. Service cooperation can be achieved via collaboration between two or more attackers, or via a centralized coordinator, *e.g.,* state-sponsored attacks. In both cases, we assume that colluding attackers have a shared, collective memory and information acquired by one attacker is available to other attacker.

Goal. The attack goal answers the question *what* attackers intend to achieve. From our case studies, we consider three goals: DDoS via distribution of malicious JavaScript, acquisition of emails, and DoS against service providers.

Technique. To achieve their goals, attackers can use different techniques. For example, law enforcement agencies may require access to user's email boxes. Network providers may intercept TCP traffic traversing their own autonomous system (AS) to inject malicious JavaScript code. In this work, we consider the following techniques: email sniffing, redirection via malicious domain resolution, in-path content injection, and hosting malicious content.

5.4 Modeling Framework

We now present our modeling framework. We base our model on *labeled property graphs*. Labeled property graphs store information in nodes and edges in the form of key-value *properties*. We present property graphs in details in Section 5.4.1. We represent elements such as domain names, IPs, organizations, and countries as nodes. Then, we use edges to represent relationships between nodes. For example, if a domain name resolves to an IP, then we add an edge between the two nodes. In a similar way, we represent relationships between IPs and countries in the sense that if an IP is located in a country, then we place an edge between the country and the IP. Finally, we use labels to specify the type of relationship.

We mine information from graphs using *graph queries*. Graph queries allow to visit graphs based on nodes, edges, and properties. In this work, we used a technique based on taint-style propagation technique and propagation rules. Starting from an initial set of nodes, we propagate a taint value according to a list of rules. Propagation rules are presented in Section 5.4.2, and queries and their evaluation are presented in Section 5.4.3.

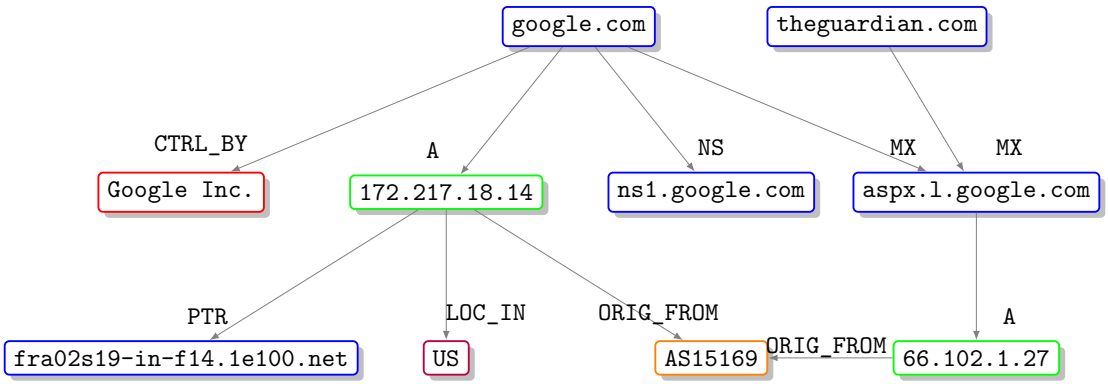


Figure 5.1: Fragment of property graph for `google.com`

5.4.1 Property Graph

A labeled property graph $G = (V, E, \lambda, \mu)$ is a directed multigraph where V is a set of nodes, $E \subseteq (V \times V)$ is a set of edges, $\lambda : V \cup E \to \Sigma$ is a function that labels nodes and edges with symbols of the alphabet Σ, and $\mu : (V \cup E) \times K \to S$ is a function that associates key-value properties, *e.g.*, (k, s) where $k \in K$ is the key and $s \in S$ is the string value, to nodes and edges.

5.1 shows an excerpt of a property graph. We use node labels to type model elements. For example, we use the label *Domain* for Internet domain names, *e.g.,* `google.com` and *Address* for IP addresses, *e.g.,* `172.217.18.14`. We use node properties to store element data. For example, we use the key *IPv4* for nodes *Address* to store the dot-decimal notation for IPv4 addresses. Our model uses other types of nodes including *Organization*, *Autonomous System* (AS), and *Country*. The full list of node labels and properties is shown in 5.1.

When we can establish a relationship between elements, we place an edge, with the label specifying the type of relationship. Relationships can be established, for example, with DNS queries and publicly available databases such as RIPE Atlas. We present data acquisition in detail in 5.5. With reference to 5.1, the domain name `google.com` resolves to `172.217.18.14` (DNS record type A), which is hosted in the AS number 15169 and geolocated in the United States. These three relationships are represented with edges labeled with `A`, `ORIGIN_FROM`, and `LOC_IN` respectively. Then, `google.com` has four authoritative DNS server one of which is `ns1.google.com`. The domain name `google.com` has also an email server `aspmx.l.google.com` whose IPv4 is `66.102.1.27`, also hosted in AS 15169. When we can also establish ownership of elements such as domain names, then we place edges between the *Organization* and the element. For example, in 5.1 we have a node `Google Inc.` which is the organization that owns the domain `google.com`. We represent this relationship with an edge `CTRL_BY`. Finally, 5.1 shows a relationship that exists between the domain `theguardian.com` and the email server `aspx.l.google.com`. The complete list of edge types is shown in 5.1.

Labels	Description
Address	Node for IP address
Domain	Node for a domain name; the source data set, *e.g.,* Alexa or JS, is a node property
DNS Zone	The zone administrated by an authoritative name server
AS	IANA number assigned to the AS; The hosted IPs is a node property
Country	Code Country code, number of IPs
Organization	Service provider name
`ORIG_FROM`	*AS* where an *Address* originates from
`LOC_IN`	*Country* where an element is located
`CTRL_BY`	*Organization* controlling, *e.g.,* a *Domain*
`A`	DNS record mapping *Address* to *Domain*
`MX`	DNS record mapping *Domain* for email delivery
`NS`	DNS record for name servers
`ZONE`	DNS record for authoritative information of a *DNS zone*
`CNAME`	Aliases from *Domain* to *Domain*
`PTR`	PTR DNS record type maps an *Address* to a *Domain*
`INCL_JS_FROM`	*Domain* name or *Address* hosting JS library

Table 5.1: Labels of nodes and relationships

5.4.2 Taint-style Propagation and Rules

A central concept of our framework is taint-style propagation and propagation rules. These elements are the building blocks to specify queries. The idea behind propagation rules is that each node of the graph may become compromised by an attacker. For example, if an attacker controls a host, then the *Address* node is considered compromised. As a consequence of this fact, *Domain* nodes that resolve to the compromised *Address* are compromised as well. The "propagation" of compromise between nodes follow specific rules that depend on the attack. Attacks may result in less severe consequences for node elements. Consider, for example, the Great Cannon attack. Web sites that included JS hosted in malicious networks can be considered compromised as well. However, in the specific case of the Great Cannon, the malicious JS code did not perform attacks against the originating server. Thus, in this case, no further entities are compromised.

Our framework supports an arbitrary granularity for compromise levels. In this work, we use three levels with the following symbols: $\mathbf{c} \in \Sigma$ for (completely) compromised, $\mathbf{pc} \in \Sigma$ for partially compromised, and $\perp \in \Sigma$ for non-compromised. When a node n is compromised (*i.e.,* $\mathbf{c}$), we add the compromise level as a node property $\mathcal{C}$, *e.g.,* $\mu(n, \mathcal{C}) = \mathbf{c}$. The propagation is implemented via rules. Each rule is a pair of preconditions and postconditions. Preconditions are evaluated on the graph. If they hold, then postconditions will hold in the graph. This is

achieved by modifying the graph such that postconditions will match. The general form of a rule is the following:

$$\frac{pre}{post} \tag{r}$$

Where *pre* and *post* are two predicates for pre and post-condition, respectively.

With reference to the previous example, the propagation rule based on the `A` (name lookup) edge is the following:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{A}}{\mu(m,\mathcal{C}) := \mathbf{c}} \tag{$r_{\texttt{A}}$}$$

This rule can be read as follows: If node n is compromised, there is an edge e between m and n, and the label of e is `A`, then we mark m as compromised.

We use similar rules to Rule $r_{\texttt{A}}$ for other type of relationships. For example, for `MX` edges we have the following rule:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{MX}}{\mu(m,\mathcal{C}) := \mathbf{c}} \tag{$r_{\texttt{MX}}$}$$

Rules $r_{\texttt{A}}$ and $r_{\texttt{MX}}$ can be applied in sequence. For example, let us assume that an address node n is compromised. Then, according to Rule $r_{\texttt{A}}$, any domain m resolving to n, *i.e.,* $(m,n) \in E$, is also compromised. If m is a domain for mail exchange, according to Rule $r_{\texttt{MX}}$, any domain p using m as its mail exchange server, *i.e.,* $(p,m) \in E$, is also compromised. In general, starting from a compromised node and a set of rules, we can propagate values $\mathbf{c}$ to other nodes.

Propagation rules are also used to represent weaker forms of compromise. Consider the case in which m is a web server hosting shared JS libraries. If m is compromised, it can, for example, distribute malicious JS libraries, which can be included in third-party websites q. As a result of this, users of q will execute the malicious code. However, this type of compromise may not entirely compromise the server of q, instead it can be used to attack other servers or compromise a user session. We model these forms of compromise with the following rule:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{JS}}{\mu(m,\mathcal{C}) := \mathbf{pc}} \tag{$r_{\texttt{JS}}$}$$

5.4.3 Query and Evaluation

We can now define more precisely a query to our model and its evaluation. A query $\mathcal{Q} = (I, R, \gamma)$ is composed of three elements: initial set of source nodes I, a set of rules $R = \{r_1, r_2, \cdots, r_n\}$, and result function γ. The set I contains nodes in G, *i.e.,* $I \subseteq N$. For example, if we want to evaluate an attack, source nodes are

the initial nodes under control of the attacker and we mark them as compromised. Then, the set R is a set of rules, starting from source nodes, that propagate the taint to other nodes. Finally, the result function γ is a generic function that given the graph G transformed by the propagation rules returns a data value.

The algorithm to evaluate a query $\mathcal{Q}$ is shown in 5.1. The algorithm is divided into three parts. The first part from 3 to 7 initializes node labels of G. Each node n in the initial set of compromised nodes I is marked accordingly, *i.e.,* $\mu(n, \mathcal{C}) := \mathbf{c}$. The remaining nodes are initialized with the symbol $\perp$. The second part of the algorithm from 10 to 16 applies the propagation rules. We use an auxiliary queue Q where we keep the rules to apply. This part of the algorithm loops over the queue until it is empty. At each iteration, we retrieve a rule r from Q, check whether the precondition holds, and apply the post conditions. The resulting graph is stored in G. If the preconditions of r still holds also in the new graph, then we enqueue r in Q. The loop terminates when Q is empty, *i.e.,* all preconditions no longer hold. Finally, we apply the result function and return the results.

Algorithm 5.1 Attack evaluation

```
1  def evaluate(G, Q)
2    # Node property initialization
3    for n in N:
4      if n in I:
5        μ(n,C) := c
6      else:
7        μ(n,C) := ⊥
8
9    # Taint Propagation
10   Q := R
11   while Q != ∅:
12     r := Q.pop()
13     if match_pre(r, G) is True:
14       G := apply_post(r, G)
15      if match_pre(r, G) is True:
16        Q.enqueue(r)
17
18   # Query result
19   return γ(G)
```

5.5 Data Sets and Acquisition

We instantiated our model on a data set of 1.8M nodes from which 350K are unique IP addresses, 1.1M are domain names and 12K are autonomous system. These nodes are connected with 4.7M relationships. Our acquisition starts from popular domains and it is expanded with server and network information. Finally, we add organizations and countries.

5.5.1 Initial Domain Names

We built our data set starting from domains that individuals and organizations may use for carrying out their daily activities. For this purpose, we used the top 100K Alexa domains, a data set of popular domain names maintained by Alexa.[3] For each domain, we created a node *Domain*. Our model contains additional domains that were implicitly acquired via Web crawling starting from the Alexa domains. To distinguish the origin of a domain name, we use a node property $\mathcal{O}$ that flags a node according to its origin.

5.5.2 Servers

Starting from the initial domain names, we resolve hosts that are responsible for core operations, *i.e.,* web servers, authoritative name servers, email servers, content distribution servers, and routers. The collection of data is done via Domain Name System queries and a Web crawler.

Authoritative Name Servers. The DNS records of a domain name are maintained by the authoritative name servers. Each authoritative name server is responsible for a portion of the domain name space, the so-called *DNS zone*. DNS zone information is stored in the SOA record type. For each domain, we retrieve the SOA record, and add a node *Zone* connected with an edge `ZONE` to the domain. Then, we retrieve the fully-qualified domain name of authoritative name servers which are listed in the NS records. For each NS record, we add a node *Domain* connected with an edge `NS` to the zone node of the domain. In addition, for each NS domain name, we resolve the IP addresses, and we add a node *Address* with the IP and an edge `A` from the domain to the IP.

Web Servers. Our initial data set is composed of domains of popular websites. By resolving the domain name, we obtain the IPs of the web server. For each of these IPs, we add a node *Address* in our model and place an edge `A` between the domain and the address. Domain names may be also have aliases via the DNS `CNAME` record. In this case, we add the alias domain in the graph and link with a

[3] `http://www.alexa.com/`

`CNAME` edge. Then, we further resolve the alias domain and add an *Address* node with `A` edge.

Email Servers. Next, we identify email transfer agents. When email clients want to send an email to a recipient, they request the `MX` record of the domain name of the email address. The `MX` record can be a list of IP addresses and domains. For each IP, we add a node *Address* and connect it with an `MX` edge to the domain. For each domain, we add a node *Domain* in the graph and the `MX` edge. Then, we resolve the domain name into an IP address and add a node *Address* with a `A` edge to the MX domain.

Content Distribution Networks. More and more websites include JS libraries that are hosted on third-party servers. For example, websites can include JS code of advertisement network services to show advertisements to their users. Websites can also use JS frameworks to support website functionalities, *e.g.,* user interface or communication with the server side. Among the popular frameworks we have, for example, jQuery and Angular.js.

Starting from a list of domain names, we identify these JS "include" relationships with a web crawler. We first visit the website and then retrieve all tags to external JS code. We also extract links to internal web pages, *e.g.,* anchor tags, and repeat the analysis on the page of the new links. We repeat this operation for a depth of 2. For each of the retrieved JS URLs, we add an *Address* node if the host is an IP, and a *Domain* node if it is a domain name. For each edge, we store the URL scheme as property using the key $\mathcal{S}$. For example, if the included JS is unprotected, *i.e.,* HTTP, then $\mathcal{S} = \texttt{HTTP}$.

5.5.3 Routing Information and Networks

We now add information about servers' networks.

Autonomous Systems—An autonomous system is a collection of IP networks and routers which are under the control of a network operator. We retrieve the origin AS of an IP using the RIPEStat database service by RIPE NCC [112]. For each AS, we create an *AS* node and add an edge from the *Address* node to the *AS* node. We additionally retrieve the total number of prefixes announced by an AS and store this number as a node property.

5.5.4 Countries and Organizations

Finally, we include countries and organization information in the graph. Our goal is to establish a relationship between these entities and the servers of 5.5.2. There are three ways to establish that, *i.e.,* at IP level, at AS level, and at domain level.

The first option is to link organizations and countries to individual IPs. This can be achieved via geolocation. Accordingly, we added geolocation data in our model using the MaxMind database [113]. While this can be achieved for countries, we are not aware of a database or an automated technique to associate a single IP to an organization controlling the server. Given the large number of IPs in our database, establishing this relationship manually is not a feasible task. The second option is to link entities to autonomous systems. This mapping is already available in RIPEstat and we include it in our model. The third is to link entities to domain names. The Domain WHOIS protocol can be used to query information about registered domain names including the domain registrant. Depending on the providing server, the structure and content of the provided information vary. WHOIS data is optimized for readability to humans [114] and thus does not have a consistent document format [115]. While a human can easily use WHOIS to retrieve data items for a single domain, it does not scale to a large volume of domain names. As an alternative source of data, we used the X.509 certificates used for HTTPS. X.509 certificates are primarily used to store servers' public-key and the domain names on which the certificate is valid. Additionally, a X.509 certificate contains the organization name to which the certificate has been issued. We included this information in our database.

5.6 Entity Identification

Before assessing attacks, we use our model to select entities that can be either attack victims or the attackers. The selection criteria are based on metrics that reflect the popularity and the influence of entities. To this end, we defined six metrics divided into *first-* and *second-order* metrics. First-order metrics are *basic* metrics which rank entities according to the number of hosted servers. Second-order metrics combine basic metrics and measure the level of influence that an entity may have on third-party services. The most popular entities of our metrics are shown in Tables 5.2 and 5.3.

5.6.1 First Order Metrics

We start with four first-order metrics, one for each server of our model, *i.e.,* name servers, web servers, email servers, and JS hosting servers. We calculate these metrics using two sets of queries, one for ASes and the other for countries.

Metric #1 (Hosted Alexa Domains). The first metric counts the number of Alexa domains hosted by an AS or a country. For an AS a, the first propagation rule is the following:

Metric 1

Country	Dom.	ASN	Name	Dom.
United States	30,582	13335	CloudFlare	7,170
Netherlands	4,296	16509	Amazon-1	2,816
Germany	4,178	14618	Amazon-2	1,892
China	4,158	20940	Akamai	1,830
Japan	3,053	16276	Ovh	1,025
France	2,526	37963	Alibaba	779
Great Britain	2,400	24940	Hetzner	725
Russia	1,678	15169	Google	525
Canada	1,186	36351	Softlayer	518
India	1,087	4134	ChinaNet	468
Ireland	986	19551	Incapsula	397
EU	950	54113	Fastly	361
Spain	848	63949	Linode	358
South Korea	755	4808	China Unic.	348

(a)

Metric 2

Country	JS	ASN	Name	JS
United States	47,910	16509	Amazon-1	10,085
Germany	7,830	13335	CloudFlare	5,489
China	7,273	20940	Akamai	3,004
Netherlands	6,963	14618	Amazon-2	2,207
Great Britain	4,455	16276	Ovh	1,970
Japan	4,205	24940	Hetzner	1,508
France	4,048	15133	EdgeCast	1,360
Russia	2,865	37963	Alibaba	940
Ireland	1,919	36351	Softlayer	910
EU	1,581	4134	ChinaNet	814
Canada	1,347	15169	Google	814
Italy	1,159	4837	China169	728
Spain	952	54994	Quantil	606
Poland	943	35415	Webzilla	551

(b)

Metric 3

Country	MX	ASN	Name	MX
United States	41,434	8075	Microsoft	8,503
Germany	12,047	16276	Ovh	2,669
Great Britain	6,811	24940	Hetzner	2,497
France	6,261	46606	Unified L.	1,353
Netherlands	6,091	36351	Softlayer	865
Japan	4,314	26496	GoDaddy	799
Russia	3,923	16509	Amazon-1	643
Italy	3,293	60781	Leaseweb	579
Canada	3,042	15169	Google	568
Ireland	2,897	39572	Advancedh.	522
Spain	2,703	12876	AS12876	452
Turkey	2,094	63949	Linode	438
Iran	1,946	14618	Amazon-2	329
India	1,892	32475	SingleHop	298

(c)

Metric 4

Country	NS	ASN	Name	NS
United States	34,235	16276	Ovh	2,415
Germany	6,697	24940	Hetzner	2,131
France	3,865	16509	Amazon	1,907
Great Britain	3,139	46606	Unified L.	1,524
Netherlands	3,116	36351	Softlayer	1,345
Canada	2,244	32475	SingleHop	1,155
Russia	2,167	13335	CloudFlare	699
Turkey	2,143	32244	Liquid Web	674
Japan	2,126	16552	Tiggee	611
Spain	1,662	26496	GoDaddy	535
China	1,617	60781	Leaseweb	398
Iran	1,552	33517	DynDNS	364
Brazil	1,070	12876	AS12876	354
India	954	4808	China Unic.	351

(d)

Table 5.2: First order metrics for identifying possible attackers and victims: (a) the number of Alexa domains, (b) number of domains hosting JS libraries, (c) number of mailexchange servers, and (d) number of name server

$$\frac{\mu(n, \mathcal{C}) = \mathbf{c}, e = (m, n) \in E, \lambda(e) = \texttt{ORIG\_FROM}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{ORIG}})$$

followed by Rule $r_{\texttt{A}}$. These two rules, starting from the source node a, propagate

Metric 5					Metric 6				
Country	**JS|NS**	**ASN**	**Name**	**JS|NS**	**Country**	**JS**	**ASN**	**Name**	**JS**
United States	41,231	16509	Amazon-1	15,429	United States	28,800	8075	Microsoft	11,596
Germany	3,101	13335	CloudFlare	4,933	Netherlands	14,213	13335	CloudFlare	6,790
Netherlands	3,045	33517	DynDNS	3,570	Ireland	11,440	16509	Amazon-1	2,018
China	3,009	4837	China169	2,008	Germany	5,380	16276	Ovh	1,969
Russia	2,254	26496	GoDaddy	1,938	Great Britain	3,116	26496	GoDaddy	1,750
France	2,084	4812	China Tlc.	1,875	France	2,996	24940	Hetzner	1,708
Japan	2,000	16552	Tiggee	1,467	Russia	2,588	33517	DynDNS	1,523
EU	1,636	16276	Ovh	1,307	Japan	1,663	36351	Softlayer	575
Great Britain	1,364	15169	Google	1,012	Spain	1,421	39572	Advancedh.	560
Spain	1,219	24940	Hetzner	873	Iran	1,123	60781	Leaseweb	478
Canada	801	15395	London Off.	822	Canada	933	16552	Tiggee	475
Singapore	787	36351	Softlayer	753	China	842	49505	Selectel	433
Poland	540	4808	China Unic.	494	Italy	798	63949	Linode	428
Iran	474	20940	Akamai	414	Turkey	797	4837	China169	352

(a) (b)

Table 5.3: Second order metrics for identifying possible attackers and victims: (a) number of JS servers whose NS is in a country/AS, and (b) number of MX servers whose NS in a country/AS

the taint value to all IP addresses and then to domain names. Domain names can originate from the Alexa database, or can be imported during the acquisition. To filter Alexa domains, we refine Rule $r_{\texttt{A}}$ by adding a check on the node property, *i.e.,* $\mu(n, \mathcal{O}) = \texttt{Alexa}$:

$$\frac{\mu(n, \mathcal{C}) = \mathbf{c}, e = (m, n) \in E, \lambda(e) = \texttt{A}, \mu(n, \mathcal{O}) = \texttt{Alexa}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{A,Alexa}})$$

Finally, we define a function γ which returns the number of compromised domains.

For a country c, we use a similar query and a new rule that propagates the taint from c to all IPs and ASes located in c. The rule is the following:

$$\frac{\mu(n, \mathcal{C}) = \mathbf{c}, e = (m, n) \in E, \lambda(e) = \texttt{LOC\_IN}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{LOC}})$$

Metric #2 (Hosted JS Libraries Providers). The second metric calculates the number of JS hosting servers which are located in an AS or a country. The approach followed is similar to the one illustrated for Metric #1, however, we use a slightly modified version of Rule $r_{\texttt{A}}$:

$$\frac{\Delta, e' = (p, m) \in E, \lambda(e') = \text{JS}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{A,JS}})$$

where Δ is the precondition of $r_{\texttt{A}}$. The new propositions $e' = (p, m) \in E$ and $\lambda(e') = \text{JS}$ describe the pattern that uniquely distinguishes JS hosting servers from other domains, *e.g.,* a domain hosts a JS program if it has an incoming edge of type `JS`.

Metric #3 (Hosted Email Servers). The third metric measures the number of email servers hosted by an attacker or victim. The query is similar to Metric #2 in which we modify Rule $r_{\texttt{A}}$ to consider domains with incoming edges of type `MX`.

Metric #4 (Hosted Name Servers). The fourth metric measures the number of name servers hosted by an attacker or victim. Also, this rule is similar to the previous ones and Rule $r_{\texttt{A}}$ consider domains with incoming edges of type `NS`.

5.6.2 Second Order Metrics

Starting from the previous metrics, we build more sophisticated ones that quantify the influence of a provider or a country on third-party servers.

Metric #5 (Name Servers for JS Providers). This metric measures the number of JS hosting servers whose authoritative name servers are hosted by a victim or attacker. The rules used for an AS are $r_{\texttt{ORIG}}$ and the following one:

$$\frac{\cdots, e = (m, n) \in E, \lambda(e) = \texttt{NS}, e' = (p, m) \in E, \lambda(e') = \texttt{JS}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{NS,JS}})$$

where we used "$\cdots$" as a place holder for the taint precondition. This rule propagates the taint from an AS to its own IPs. An IP is counted if two conditions are met. First the IP n has an incoming edge `NS` from another node m, *i.e., n* is an authoritative server for m. Second, the node m has an incoming edge of type `JS` from a node p, *i.e., m* hosts a JS library for p. The query for the case of a country contains the Rule $r_{\texttt{LOC}}$ followed by $r_{\texttt{ORIG}}$ and $r_{\texttt{NS,JS}}$.

Metric #6 (Name Servers for Email Servers). This metric measures the number of domain of email servers whose name server is hosted by a victim/attacker. The construction of the query is the same as for Metric #5. In the case of AS, the rules used are $r_{\texttt{ORIG}}$ and a modified version of $r_{\texttt{NS,JS}}$:

$$\frac{\cdots, e = (m, n) \in E, \lambda(e) = \texttt{NS}, e' = (p, m) \in E, \lambda(e') = \texttt{MX}}{\mu(m, \mathcal{C}) := \mathbf{c}} \qquad (r_{\texttt{NS,MX}})$$

5.7 Attack Evaluation

We now evaluate the impact of attacks. We consider three attack scenarios, namely, distribution of JS malicious content Section (5.7.1), email sniffing Section (5.7.2), and DoS against core service providers Section (5.7.3). We present results with two levels of granularity. First, we show the overall impact of attacks in terms of total number of affected Alexa domains. Second, for a selection of attacks, we present attack results on a per-victim base.

Country	Host Collusion	In-path Injection	ASN	Name	Host Collusion	In-path Injection
United States	15,658	12,267	15169	Google	9,469	5,553
Netherlands	3,292	2,639	13335	CloudFlare	4,310	3,165
Russia	1,701	1,409	15133	EdgeCast	3,404	2,306
Germany	1,622	1,317	16509	Amazon-1	3,216	2,264
Japan	1,311	1,151	20940	Akamai	2,279	1,800
China	1,141	1,079	14618	Amazon-2	572	351
Great Britain	1,094	895	35415	Webzilla	515	479
Ireland	1,048	828	24940	Hetzner	379	330
EU	905	824	16276	Ovh	342	287
France	713	603	36351	Softlayer	334	286
Canada	399	246	4837	China169	227	226
Poland	176	151	4134	ChinaNet	198	185
Italy	105	97	37963	Alibaba	148	146
Spain	83	69	54994	Quantil	71	71

Table 5.4: Attack evaluation: Distribution of malicious JS content with hosting malicious JS content and in-path malicious JS injection

5.7.1 Distribution of JS Malicious Content

For this attack, we consider three techniques: hosting malicious JS content, injection of malicious JS on in-path TCP connections, and malicious name resolution redirection. We select attackers according to metrics #2 and #4 in Table 5.2. Then, for each technique and attacker, we measure attack results as the number of websites that, as a result of the attack, will distribute the malicious JS content to their users. Table 5.4 and Table 5.5(a) show the attack results when the attacker is an AS or a country, respectively.

Hosting Malicious JS Content. In this attack we assume that the attacker is either an AS or a country that colluded with web servers hosting JS code. For example, in the case of AS, we assume that the web servers hosted by the AS are cooperating with the origin AS. Possible attackers can be selected with Metric #2, which count the total number of domains hosting JS for each AS or country.

Country	DNS redir.	ASN	Name	DNS redir.
United States	12,375	15169	Google	7,859
Russia	1,362	33517	DynDNS	4,311
Netherlands	1,225	16509	Amazon-1	3,685
China	1,032	13335	CloudFlare	3,012
Japan	880	4812	China Tlc	595
EU	743	4837	China169	555
Germany	621	4808	China Unic	401
France	454	16552	Tiggee	361
Singapore	317	26496	GoDaddy	316
Great Britain	225	24940	Hetzner	227
Spain	173	16276	Ovh	199
Iran	124	36351	Softlayer	196
Canada	117	20940	Akamai	88
Poland	65	15395	London Off	88

(a)

Country	MX Coll.	ASN	Name	MX Coll.
United States	24,459	15169	Google	11,127
Germany	2,301	8075	Microsoft	2,465
Great Britain	1,838	26496	GoDaddy	1,267
Russia	1,602	16276	Ovh	565
France	1,382	24940	Hetzner	347
Japan	1,317	16509	Amazon-1	332
Netherlands	1,279	36351	Softlayer	237
Ireland	809	60781	Leaseweb	170
Canada	614	12876	AS12876	134
India	496	46606	Unified L	113
Spain	410	63949	Linode	108
Iran	392	14618	Amazon-2	104
Italy	384	39572	Advancedh	96
Turkey	319	32475	SingleHop	93

(b)

Country	MX+NS	ASN	Name	MX+NS
United States	13,077	8075	Microsoft	3,003
Netherlands	3,933	13335	CloudFlare	2,280
Ireland	3,006	4837	China169	1,784
China	2,300	26496	GoDaddy	1,447
Germany	1,405	16509	Amazon-1	1,256
Great Britain	1,735	33517	DynDNS	1,178
Russia	1,466	16276	Ovh	555
France	910	24940	Hetzner	335
Japan	902	16552	Tiggee	227
Iran	344	36351	Softlayer	179
Spain	338	39572	Advancedh.	96
Canada	265	60781	Leaseweb	75
Italy	242	49505	Selectel	65
Turkey	213	63949	Linode	57

(c)

Table 5.5: Attack evaluation: (a) malicious name resolution (b) email sniffing via malicious email provider, and (c) malicious name resolution for email sniffing

The attack results are shown in Table 5.5. The attack results show that countries can be very powerful attackers. For example, according to Metric #2, the United States hosts 47K JS hosting providers (see Table 5.2(b)) which could distribute malicious code to about 16% of the top 100K Alexa domains. However, ASes are also very powerful and affect a fraction of websites that is even larger than that of individual countries, and even groups of countries. For example, the AS of Google can affect about 9% of Alexa domains, the number of domains

that can be affected by the Netherlands, Russia, Germany, Japan, China, and Great Britain combined. Even more interestingly, the AS of Google reaches 9% of websites with only 762 servers compared to 3% of the 10K servers of Amazon. This result highlights that the power of operators can be more precisely measured by taking into account to what extent other services depend on them. The AS of Google is not an isolated case. Other ASes can affect as many domains as a country. Examples of these ASes are CloudFlare EdgeCast, Amazon-1, and Akamai. Each of them can distribute malicious code to more domains than the top six countries (excluding the United States).

Propagation rules—We created this table with the following rules. When the attacker is the AS, we use Rule $r_{\texttt{ORIG}}$, $r_{\texttt{A}}$, and $r_{\texttt{JS}}$. If the attacker is a country, then we use the Rule $r_{\texttt{LOC}}$ followed by the previous ones, *i.e.,* $r_{\texttt{ORIG}}$, $r_{\texttt{A}}$, and $r_{\texttt{JS}}$. The resulting graph is then processed by the γ function, which counts the number of tainted Alexa domains.

In-path Malicious JS Injection. Interestingly, a very large fraction, *i.e.,* 82% (Table 5.4), of JS hosting service distribute JS libraries over unprotected connections, *i.e.,* HTTP instead of HTTPS. Accordingly, hosting ASes and countries can intercept TCP connections from border gateways and inject malicious content similarly as performed for the Great Cannon attack. We may extend the measurement to protected resources, however, the attacker is required to control a valid certificate for the domain being hijacked. While this is a possible attack scenario, it requires additional effort that, considering the low number of protected resources, will produce a limited increase of the attack result. Table 5.4 shows the attack results on Alexa Web sites that include an unprotected JS program.

Among the 82% of JS inclusion over unprotected connections, 1,079 of them are crossing the Chinese network borders. However, China is not the country that can affect the largest fraction of websites. Other countries could perform better than China including the United States with 12,267 websites, the Netherlands with 2,639 websites, and Russia with 1,409 websites. An interesting aspect of our results is that this type of attack method does not perform any better than the hosting malicious content attack. In fact, injecting malicious JS code via web server collusion affects 17% fewer affected domains on average than hosting malicious content.

Similarly to the attack based on hosting malicious content, we observed that ASes can affect more domains than countries. For example, the AS of Google can affect as many domains as the Netherlands, Russia, and Germany together. However, also in this case, in-path malicious JS injection does not reach as many domains as the injection via server collusion. For example, an in-path code injection can cause Google to lose about 41% of total websites.

Now, we present a fine-grained analysis of this attack. We map the attack results to countries that would be affected if another country decides to perform

this attack. An excerpt of these results are presented in Table 5.6. Attack results can be interpreted as a form of dependency among countries. Our results show two interesting facts. First, with different intensity, almost all the popular countries (except for six of them) can attack at least one domain of another country. Second, the dependency among countries is not symmetric. For example, consider the United States. According to all metrics, the United States is the most powerful attacker in our model. However, this influence is not symmetric, *e.g.,* when compared to the Netherlands. While the United States can affect 283 Dutch domains, 967 US domains can be attacked by the Netherlands.

Propagation rules—The rules for this measurement are similar to those of the previous attack. However, we modified Rule $r_{\texttt{JS}}$ to limit the propagation to unprotected JS edges only:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{JS}, \mu(e,\mathcal{S}) = \texttt{HTTP}}{\mu(m,\mathcal{C}) := \mathbf{pc}} \qquad (r_{\texttt{JS}})$$

where $\mathcal{S}$ is the property key that stores the URL scheme of the include JS.

Malicious Name Resolution Redirection. Finally, malicious content can be distributed to Web browsers via malicious domain name resolution. In this attack, we assume that the authoritative name server of a domain hosting JS redirects users to a malicious server. The attack result is the number of websites that include a resource hosted on a server whose name server is colluded or compromised. This attack exploits three types of relationships of our model. The first relationship is between domains and the domains hosting JavaScript. The second relationship is the domain name resolution which maps, eventually, domain names to IPs. The third type of relationship is the domain name resolution process with an operator, *e.g.,* country or network provider.

With this technique, countries do not gain considerably more power than the previous attacks. In most of the cases, all players can affect a slightly lower number of websites. Only two players stand out from the rest, *i.e.,* the AS of Google and DynDNS. Google hosts 73 name servers that can be used to distribute malicious JS to the users of 7,859 domains. This amounts to an increase of 677% of the number of controlled name servers. Similarly, DynDNS controls 3,570 name servers that can affect 4,311 domains. We discuss in detail the role played by DynDNS in our model in Section 5.7.3.

Propagation rules—We create this table with the following rules. When the attacker is the AS, we use Rule $r_{\texttt{ORIG}}$, $r_{\texttt{A}}$, and $r_{\texttt{JS}}$. If the attacker is a country, then we use the Rule $r_{\texttt{LOC}}$ followed by the previous ones, *i.e.,* $r_{\texttt{ORIG}}$, $r_{\texttt{A}}$, and $r_{\texttt{JS}}$. The resulting graph is then processed by a γ function that counts the number of tainted Alexa domains. The number of affected domains is showed in Table 5.5(a). The used propagation rules are $r_{\texttt{ORIG}}$, $r_{\texttt{A}}$, $r_{\texttt{Ns}}$ and $r_{\texttt{JS}}$, where $r_{\texttt{Ns}}$ is defined as follows:

$$\frac{\mu(n,\mathcal{C}) = \mathbf{c}, e = (m,n) \in E, \lambda(e) = \texttt{NS}}{\mu(m,\mathcal{C}) := \mathbf{c}} \quad (r_{\texttt{Ns}})$$

5.7.2 Email Sniffing

To acquire a large number of emails, an attacker can rely on various techniques. In this work we consider two. The first one is by acquiring them directly from the email server. The second one is by redirecting an email client toward a malicious mail server, which will accept the email, keep a copy, and forward it to the intended recipient. This attack can be performed by a provider or by a country. Tables 5.5(b) and 5.5(c) show the attack results. All values are the number of Alexa domains that will be affected by this attack grouped by technique and attacker.

Malicious Email Provider. Attackers that can perform this type of attack are selected using Metric #3, *i.e.,* ASes or countries hosting email servers. This attack technique shows the predominance of the United States and Google in managing the email infrastructure of a large fraction of popular websites (See Table 5.5(b)). The United States alone can acquire emails of 25% of the most popular websites. Similarly, the AS of Google is hosting only 568 email servers which are used by 11% of the websites. The other players, such as Germany, have still relevant influence but up to 10 times less than the US or Google. Interestingly, most of the domains that can be affected by a country are hosted in the same country. For example 17K of the domains affected by the US are hosted in the US (See Table 5.7).

Malicious Name Resolution Redirection. Attackers for this attack are selected using Metric #6. This metric measures the number of mail server whose authoritative name server is hosted by the attacker. Starting from this list of attackers, we counted the number of Alexa domains that use one of these mail servers. Table 5.5(c) shows the total number of affected Alexa domains. As we cannot measure to what extent TLS is used as part of the client-to-MTA or MTA-to-MTA email transfer, we neglect the fact that malicious name servers potentially cannot redirect communication. The numbers provided in this scenario thus rather constitute upper bounds.

With this type of attack, we observe that Google loses most of its power. This can be explained by the fact that websites use Google email server via name servers which are not hosted by Google.

5.7.3 DoS against Core Service Provider

In this section, we consider the case in which a service provider is the victim of an attack. Here, we do not focus on the specific attack technique, but on the impact of making a provider unavailable. The metrics of Tables 5.2 and 5.3 can be used to select a candidate. The queries used for the attack results can be reused for this type of assessment.

For example, let us consider the DoS attack that on the 21st of October 2016 was launched against Dyn.com. DynDNS is an autonomous system operated by Dyn.com. According to our model based on the top 100K Alexa domains, DynDNS does not host a relevant number of mail servers and JS hosting providers. However, it hosts 364 domain name servers. These name servers are authoritative for 3,570 (see Table 5.3 (a)) domains hosting JS that provide JS to 5,559 top 100K Alexa domains (not shown in Table 5.4), of which 4,331 are unprotected JS inclusion. Furthermore, the name servers hosted by DynDNS are authoritative for 1,523 domains running mail servers which are used by 1,178 top Alexa domains. If the Dyn.com DNS infrastructure is attacked, then a fraction that ranges from 1% to 5% of the top 100K Alexa domains would be affected. The operation of these domains may be severely compromised, as JS used to deliver services via Web applications would no longer be available.

	CN	**DE**	**FR**	**GB**	**JP**	**NL**	**RU**	**US**	**Total**
US	134	355	223	172	657	290	287	4,248	6,366
RU	0	107	22	14	0	70	364	124	701
NL	12	53	42	42	189	171	55	979	1,543
JP	7	4	1	1	597	7	0	185	802
GB	79	20	15	49	21	29	9	322	544
FR	0	18	84	10	24	10	17	150	313
DE	77	191	31	14	33	38	62	312	758
CN	475	6	3	0	19	0	1	109	613
Total	784	754	421	302	1,540	615	795	6,429	11,640

Table 5.6: JS injection on in-path TCP connections group by countries.

5.8 Limitations

We evaluate the impact of the attacks over static data acquired in a single time point, which can be seen as a snapshot of the current network status. Therefore the dependency graph is static. Also, we did not consider different network views (*e.g.,* from different locations) and the data has been collected from a single vantage point located at Saarland University in Germany. This vantage points may have an influence especially for geographically distributed Content Delivery

	CN	DE	FR	GB	JP	NL	RU	US	Total
US	1,472	1,060	626	765	455	2,047	229	17,245	23,899
RU	0	179	53	24	0	72	933	268	1,529
NL	5	85	43	72	0	512	21	548	1,286
JP	9	13	0	3	1,149	59	0	112	1,345
GB	9	165	115	766	59	292	16	496	1,918
FR	1	40	918	50	1	66	5	356	1,437
DE	6	1,503	88	144	4	209	37	542	2,533
CN	2,387	31	3	6	18	13	0	281	2,739
Total	3,889	3,076	1,846	1,830	1,686	3,270	1,241	19,848	36,686

Table 5.7: Attack results of malicious email providers grouped by countries

Networks (CDNs), in that our analyses are potentially biased based on the client's residence. An interesting direction for future work would be to include different vantage points. Besides geographic differences based on the Web topology, we believe that building a dependency graph from different vantage points especially from countries deploying censorship filters could also reveal other intriguing results and change the impact of the attacks.

To acquire JavaScript dependencies between domains, we used Scrapy[4], a framework to extract static data from websites. Scrapy does not support the execution of JavaScript. As a result, our model does not include dependencies that may originate from the execution of the JavaScript program. The extraction of these dependencies can be achieved by using advanced Web crawlers (*e.g.,* jÄk [116]). While this, as shown by prior research, can increase coverage up to 80% [116], it is not a viable option for large-scale analyses. Crawlers that interpret JavaScript are considerably slower than classic crawlers as they require more resources and longer execution time for each website.

5.9 Related Work

The security of the Internet infrastructure has been under constant scrutiny of the research community. Countless works have been presented by using formal and empirical analyses. For example, Albert *et al.* [101] studied Internet robustness against random errors and targeted attacks. They show that the Internet provides high error tolerance, but it does not provide adequate robustness against attacks targeting hubs (*i.e.,* nodes with higher connection degree). Following this seminal work, other works assessed other aspects of the problem such as hub selection (*e.g.,* [102, 117]). Following this research line, our work takes an empirical approach to mine topology of the Internet infrastructures to study the impact of large-scale attacks.

[4]See `https://scrapy.org/`

The security of the Internet has been studied also empirically with measurements of BGP infrastructure [107], JS inclusion [118], Web service networks [106], and HTTPS ecosystem [119]. Frey *et al.* [107] presented an analysis on the European BGP backbone using publicly available BGP data. Nikiforakis et al. [118] showed that the vast majority of websites rely on external JS libraries stored on poorly maintained web servers. Finally, Cangialosi *et al.* [119] studied dependences among providers based on shared X.509 certificates, and its implications on the HTTPS ecosystem security. Our work presents a similar *what-if* analysis which complements these papers. However, while these works considered individual service in isolation, our work is more comprehensive, considering different services, intra-services relationships, and a framework to support analyses similar to the aforementioned works.

Classical data mining approaches have been applied to security problems as well. Data mining has been used to classify system and user behaviors to detect outliers [120]. Mining techniques have been used also to detect vulnerabilities. For example, source code can be represented as a property graph, and graph traversals can be used to identify vulnerable behaviors [121, 122]. Unlike all these works, that are mainly focused on the analysis of a single protocol, network service, or network layer, our methodology allows to study the security of the infrastructure considering the various services involved, organizations, and dependencies between them.

Finally, another line of works attempts to learn service dependencies via observations of network traffic (*e.g.,* NSDMiner [123] and Rippler [124]). While these tools can effectively learn dependencies, they require network traffic which is not available for the global analyses like the one presented by our and other works (*e.g.,* [119]).

5.10 Conclusion

In this chapter, we proposed investigation techniques to assess global-scale threats. We presented a model of the Internet infrastructures based on property graphs. Moreover, to mine the data from the model, we presented a taint-style propagation technique for traversing the graph. We evaluated our framework, on a model built upon the top 100K Alexa domains by passively and actively collecting publicly available information. Using the presented metrics for selecting attacker and victim candidates, we assessed the impact of the attackers and identified the most influential Internet players. Finally, we showed how one country can influence another by using JS injection on in-path TCP connections and MX server collusion.

6

TTL-based Filtering

On the Feasibility of TTL-based Filtering for DRDoS Mitigation

6.1 Motivation

One of the major hassles for network provides in recent years have been so-called *Distributed Reflective Denial-of-Service* (DRDoS) attacks [125]. In these attacks, an attacker poses as its victim and sends a flood of tiny packets to vulnerable services which then respond with much larger replies to the victim. This is possible because the Internet Protocol (IP) does not have means to protect against forgery of source addresses in its packets, so-called *IP spoofing*. A variety of different UDP-based protocols have been known to be vulnerable for this category of attacks for long [126], but despite the efforts to locate and shut down vulnerable services, they remain a problem even today.

To ensure that a server does not become unwilling participant in a DRDoS attack, an appealing defense is to detect spoofed packets *at the recipient*. One such technique is to validate certain IP header fields and drop packets that seem unsound. Most promising, Cheng *et al.* [127] propose a technique called Hop Count Filtering (HCF) to leverage the Time-to-Live (TTL) field encoded in the IP header. The intuition behind a TTL-based filtering approach is that the route of the *actual* source of the traffic and the *claimed* source is likely different, *i.e.,* the spoofing source is in a different network than the spoofed IP address. This is then also reflected in the TTL value, as the attacker's route to the server differs from the one of the spoofed system, and hence the number of hops is different. Thus, it is seemingly possible to filter most spoofed traffic by dropping any traffic which does not correspond to the expected TTL.

6.2 Contribution

In this work, we evaluate the feasibility of using HCF to defend against DRDoS attacks. To do so, we analyze several means of probing for the TTL of an alleged sender, using different types of probes towards a host in question as well as horizontal probing of its neighbors. We show that this process is prone to errors and frequently tedious in practice, raising the need for a certain tolerance in TTL-based defenses. More precisely, we show that an error margin of +/-2 must be allowed to enable 86.4% of benign traffic to pass, while dropping more than 75% of spoofed traffic.

Any TTL-based defense relies on the tacit assumption that an attacker cannot learn the correct TTL when spoofing a packet. We, however, show that a spoofing attacker can subvert TTL-based filters by predicting the TTL value—without having access to the system or network of either server or impersonated victim. Our idea is to leverage publicly available traceroute data to learn subpaths that an IP packet from IP_A to IP_B will take. We follow the intuition that subpaths from IP_A to any other host on the Internet are quite constant and can be learned

by the attacker. Similarly, we show that the attacker can observe that any packet to IP_B traverses a certain subpath. We augment such subpath information with an approximation of how the packet is routed on the higher-tier Internet layers. Given the tolerance required in TTL-based defenses, we can estimate the initial TTL value that the attacker has to set to enable bypassing of such defenses. These "negative" results prove that TTL-based spoofing filters are unreliable and (if at all) a short-sighted solution only. Rather than attacking existing defense systems, our findings conceptually show that TTL-based defenses cannot work to thwart the outlined attacks. Hence, we see this work as a valuable contribution to steer future research towards more fundamental solutions, be it alternative defenses against spoofing, or conceptual redesigns of the Internet and its protocols. To summarize, we make the following contributions:

- We re-evaluate the concept of HCF to determine the necessary level of tolerance required for it to work in practice.
- We describe a methodology which leverages previous knowledge about routing and statistical models to estimate the number of hops between an arbitrary victim and an amplifier of the attacker's choosing.
- In doing so, we show that TTL-based defenses can be circumvented by an attacker with as little as 40 globally distributed probes.

6.3 Background

In this section, we discuss the background information on routing on the Internet, Distributed Denial of Service attacks, and Hop Count Filtering as a countermeasure against such attacks.

6.3.1 Relevant Internet Technologies

The Internet is a globally connected network system that interconnects sub-networks, which route packets between them based on the established routes. These smaller networks, also referred to as *Autonomous Systems* (AS), are collection of IP routing prefixes under a control of a single administrative entity with internally defined routing policies.

For a host in network A to connect to a host in network B, a route must be found through potentially several different ASes. Exchanging traffic between different autonomous systems is routed based on the Border Gateway Protocol (BGP), in which routers exchange information about accessible IP ranges and the corresponding AS paths, *i.e.,* routes to these ranges.

To ensure that a packet is not stuck in a routing loop, the Internet Protocol (IP) header contains a field named Time-to-Live (TTL). When handling a packet, "[...]

every module that processes a datagram must decrease the TTL" and whenever a packet's TTL value reached zero, the packet must be discarded by the routing device [128]. In practice, the TTL is a very simple concept implemented as a decreasing hop count. The value is initially set by the sending host and it usually depends on the operating system, *e.g.,* Mac OS X uses 64, Windows 128, and while Linux distributions nowadays mostly use 64, some even use 255 [129]. When receiving a packet, analysis of the TTL values therefore allows to approximate the number of routing devices the packet has passed.

The concept of TTLs can also be used to learn the exact route of a packet (*tracerouting*). To that end, the initiator of the tracerouting sends an IP packet towards the intended destination, initially setting the TTL value to 1. When this packet reaches the first hop, the TTL is decreased. According to the RFC, the router must now drop the packet. In such a case, most routers will also send an Internet Control Message Protocol (*ICMP*) error message to the original sender, indicating that the timeout of the packet has been exceeded. This response can be used by the tracerouting machine to learn the IP address of the first hop. By repeating this process with increasing TTL values, this method can be used to learn all IP addresses of routers on the packet's way to its destination.

6.3.2 Source Spoofing and DRDoS

By design, the Internet Protocol does not feature a means of verifying the source of a packet. Since IP packets are only directed based on the *destination*, an attacker can easily generate an IP packet with a fabricated (or *spoofed*) source address. This drawback of the original design of the Internet Protocol gives the adversary plenty of space for misusing it towards several aims. One example is Denial of Service (DoS) attacks, where an attacker tries to either saturate the network link to a server or exhaust resources on the target machine by, *e.g.,* initiating a large number of TCP handshakes. To defend against this, a network administrator may configure a firewall to drop packets from the attacker. The attacker, however, can easily bypass this defense mechanism by spoofing the IP packets making them appear to come from different addresses.

Moreover, recent years have seen an increase in Distributed Reflective Denial of Service (DRDoS) attacks. These attacks rely on spoofing packets in conjunction with services which respond to requests with significantly larger responses. There are a variety of vulnerable protocols (described in [126, 130]), but recently, the most nefarious attacks have been misusing protocols such as DNS, NTP, SSDP, or chargen. As an example, the Network Time Protocol's (NTP) *monlist* feature may generate a response that is more than 4,500 times larger than the request. To abuse this, an attacker generates a flood of *monlist* requests to vulnerable servers while spoofing the source IP address to be that of the victim. Subsequently, a vulnerable NTP server will send the response to the victim's IP. As a result, the

attacker massively amplifies his own bandwidth, and at the same time hiding his real IP address.

Although this kind of attack has been well-known for long [131, 132] and attempts have been made to shut down vulnerable systems used in such attacks (*e.g.,* [133]), they still pose a threat to online services. In order to fight such attacks, several countermeasures dating back to 2001 [134] have been proposed. One obvious defense strategy would be to limit the number of requests a client may issue. However, while such mechanisms may help to protect against excessive abuse of a single amplifier, Rossow's [126] analysis shows that even with rate limiting the aggregated attack bandwidth of some protocols is still an issue. This and many other countermeasures have been evaluated and analyzed by Beitollahi and Deconinck [135], hence we omit to discuss them further and refer the reader to their paper. Instead, we discuss the hop count filtering mechanisms relevant for our work in the following.

6.3.3 Hop Count Filtering

When a packet is received, its TTL depends on (i) the initial TTL value and (ii) the number of hops the packet has traversed. While it is easy to forge an IP header as such, Cheng *et al.* [127] propose to use the TTL to detect nefarious packets. More precisely, they assume that an attacker trying to impersonate a specific host cannot ascertain the hop count between the spoofed host and the recipient of the packet. Based on this assumption, they present a reactive defense against DDoS attacks. To detect an attack in which the sender spoofs IP addresses to conceal his true location, they first require a period of observing the legitimate upcoming traffic (learning state), where the victim builds a mapping between the legitimate clients (IP addresses) and their respective hop count. Once an attack is detected, the victim rejects all packets where the TTL values do not match the recorded hop count. This way, the victim does not have to allocate resources for handling incoming spoofed traffic.

To increase the accuracy of the hop count filtering (HCF), Mukaddam *et al.* [136] proposed a modified version of HCF that aims at improving the learning phase. Instead of recording only one hop count value per IP, they record a list of all possible hop count values seen in the past. They justify the need for such an extension by arguing that the hop count may change due to the use of different routes. Indeed, such a system decreases the collateral damage by correctly classifying legitimate traffic. On the other hand, however, this mechanism allows an attacker to make more guesses in evasion attempts by ascertaining the correct TTL value.

6.4 Re-Evaluating the Feasibility of Hop-Count Filtering

The work by Mukaddam *et al.* has shown, that the efficiency of the original HCF approach may be reduced by the nature of routing on the Internet. In addition, such an approach requires a prior learning phase, *e.g.,* through passive TCP handshake analysis, to facilitate detection of spoofing. In the following, we investigate how far the methodology from Cheng *et al.* can be extended to filter out spoofed traffic used in DRDoS attacks. In contrast to the original HCF, this process cannot rely solely on TCP handshakes from previous connections by the client, as protocols used in DRDoS attacks, such as NTP or DNS, are connection-less. Simply dropping all packets from any host without a previous TCP connection would render any benign use of UDP-based services moot. Therefore, we investigate what is the error margin of the TTL value for an alleged sender that can be learned by the server to evaluate the efficacy of TTL-based filtering on the Internet.

6.4.1 Protocol-based Probing

The most intuitive way for a server to ascertain a TTL value of a client is to receive a genuine (non-spoofed) packet from that host. This can be done after a successful TCP handshake, as an established connection can only occur if the alleged sender actually initiated the connection. Due to its connection-less nature, we cannot rely on such a process for UDP. Instead, we need to prompt the alleged sender for a non-spoofed packet. To achieve this, we can rely on ICMP, TCP, or UDP requests to the system in question. The ports we used in our work for TCP and UDP are derived from the most scanned port discussed by Durumeric *et al.* [137]. We realize that it might not be feasible to send a plethora of probes to an end host whenever a packet to a UDP-based service is received, as this itself would be an amplification attack. Regardless, we want to investigate how different protocols and techniques might be leveraged to learn the TTL.

One way of compelling the probed system to send a packet is to use ICMP. ICMP *echo* can be used to measure the round trip time of a packet to a given host. The TTL of the probe target can be extracted from the IP header of an echo reply. In addition to the echo command, several operating systems also implement the non-mandated *timestamp* command. This can be used in the same fashion to induce a response from the probed system.

Additionally, the probing server can itself try to establish a TCP connection to the alleged sender. The methodology is independent of the actual application used underneath, since the TCP handshake is conducted by the operating system before handing the socket to the underlying application.

In contrast to TCP, where no application data needs to be sent to the probed

host, most UDP-based services require protocol-specific data to be submitted. As an example, DNS and NTP servers only react to datagrams which are conformant to the respective protocol. On the other hand, the UDP-based *chargen* service "simply sends data without regard to the input" [138]. Therefore, we send protocol-conformant packets to DNS and NTP ports, and random data to chargen.

6.4.2 Interpreting Responses.

In any of the cases described above, we may receive a positive or negative response. In the following, we discuss these types of responses and indicate how they can be used to extract the TTL from probed systems.

Positive Responses. When using ICMP, an echo or timestamp reply suffices to extract the TTL value from the encapsulating IP packet. For TCP, if a service listens on the probed port, the operating system will follow the three-way handshake process and respond with a SYN/ACK packet. In the case of UDP, the process differs slightly: when a service is listening on the probed port and the incoming packet adheres to the specification of that service, it sends a response back to the requesting system. Analogously to ICMP, the TTL value can be extracted from TCP and UDP responses by simply examining the IP header.

Negative Responses. In addition to responses which indicate that the host is up or a service is listening on the probed port, we can also leverage negative responses or error messages to learn the TTL. For example, in cases where a TCP port is not open, the host system may respond with a packet which has the RST flag set. Assuming that the packet is usually generated by the probed system (we discuss exceptions to this rule in Section 6.4.4), we can extract the TTL value in the same fashion used for positive responses. For UDP, we leverage ICMP *Port Unreachable* replies.

Next to these protocol-specific errors, we may also receive a message indicating that the host is not reachable. For example, the last router on the path can issue an ICMP *Host Unreachable* message. In this case, given the assumption that only the last router will send such a message, we can use the TTL from the incoming packet and decrease it by one (since the original sender would have had one more hop). ICMP also features a more generic *Destination Unreachable* message; this, however, can be sent by any router on the path and therefore cannot be used to conclusively calculate the TTL value. Next to these, we may receive ICMP *Communication Administratively Prohibited* messages. Such a message can either be sent by a router or the system itself when a packet is rejected by the firewall.

6.4.3 Horizontal Probing

A probed host may not answer, *e.g.,* because it is behind a firewall and drops any incoming packets. In these cases, we may still gather valuable information on the path to the host by probing neighboring hosts. A neighbor in this case is a host which is located within the same subnet as the target. Although assuming that each subnet consists of exactly 256 IPs is not correct, this measure can still provide partial insight into the route and give a close estimate of the actual TTL value. Therefore, we probe neighbors by changing the last octet of the IP address +/-1, and use previous knowledge from hosts within the same /24 subnet, as this is the smallest network section generally advertised and accepted via BGP [139].

6.4.4 Caveats of Active Probing

There are several scenarios which can induce errors in probes. Typically, private customers receive a router for their dial-up account, which uses Network Address Translation (*NAT*) to allow multiple LAN clients access to a single Internet connection. Unless these routers are configured to forward packets to a machine behind the NAT, any response to the previously mentioned probes will be generated by the router. As the router adds an additional hop (and hence decreases the TTL by one) on the way from the NAT client to the server, the TTL values will mismatch in such a case.

For negative responses, additional artifacts may skew the results. Specifically, TCP resets or ICMP error packets may be generated by a firewall located before the intended probe target. In such a case, the firewall itself must spoof the probed IP to send these packets to ensure that the packet is attributed correctly on the system which initiated the connection. Hence, we may assume that negative responses are indeed generated by the probed system. Since we cannot learn the number of hops between the firewall and probed system, using negative responses can yield false results. We discuss the number of false results in Section 6.5.

As outlined before, the initial TTL value depends on the operating system of the sending host. Considering an example in which a Windows client is located behind a NAT router, which is running a Linux system with an initial TTL value of 255. Even though a packet originating from the Windows machine will only have one additional hop on its way to the probing server, the TTL value received by the probing system will greatly differ depending on whether the Windows or Linux host responded to the probing request. To accommodate for this and for horizontal probing, we normalize all TTL values to values between 0 and 63, *i.e.,* $TTL = TTL\%64$. As the maximum TTL of 255 is not divisible by 64, we first increment TTL values above 128 by one to correct this discrepancy.

6.5 Probing Analysis

To evaluate how well active probing could be used in the wild to enable the use of HCF, we set up two systems. First, we used a regular NTP server not susceptible to DRDoS to attract benign clients. Second, we set up a honeypot system running a vulnerable version of NTP to attract spoofing attackers. In the following, we describe both data sets, discussing for what fraction of hosts we could learn any TTL value, and comparing this to the TTL values of incoming packets. Although we are using NTP servers for our evaluation, it is out of convenience of getting both spoofed and non-spoofed clients for comparison. In contrast, for protocols like chargen, getting benign traffic would have been significantly harder. We end this section with a discussion on the implications of the results of our analysis.

6.5.1 Benign Traffic

To capture benign traffic, we set up an NTP server that does not implement *monlist* feature at all, and is therefore not susceptible to amplification vectors. To attract NTP clients, we joined the NTP pool project. Note that the term *client* refers to its role in NTP, *i.e.,* such a host could either be an end user's computer or a server synchronizing its clock with us. Within hours, the server was added to the public pool and started to receive NTP requests. We analyzed the incoming traffic for patterns of suspicious behavior (especially dreaded *monlist* requests). Our analysis showed that such requests were only issued in small numbers by scanners (*e.g.,* operated by research groups). As we did not respond to such amplification requests and did not notice any suspicious activity, it is highly unlikely that an attacker would choose our server for his amplification attack. Hence, we deem this data set to consist exclusively of unspoofed traffic.

In total, we gathered data for 48 hours, in which we received packets from 543,514 distinct IP addresses. In a first step, we probed each of these hosts immediately after their first contact using the different types of probes outlined in Section 6.4.1. In doing so, we could extract TTL values for 316,012 (58.1%) for probed systems. The most successful type of probe was ICMP echo, which yielded a result for 257,694 or 47.4% of the hosts. In comparison, the most successful TCP-based, positive response were SYN/ACKs from TCP port 443 (HTTPS), which accounted for a mere 31,966 (5.9%) of the hosts. For any UDP-based probes, we only received negligible amounts of positive responses. Among the negative responses, ICMP *Communication Prohibited* for TCP port 4899 (Radmin) was the most frequent message (113,058 or 20,8%).

To find out how accurate these results actually are, we compared the normalized TTL values to the ones from the incoming traffic. As stated before, we assume that the traffic directed to the NTP server is indeed generated by the alleged senders, *i.e.,* the ground truth value for each sending host can be extracted from

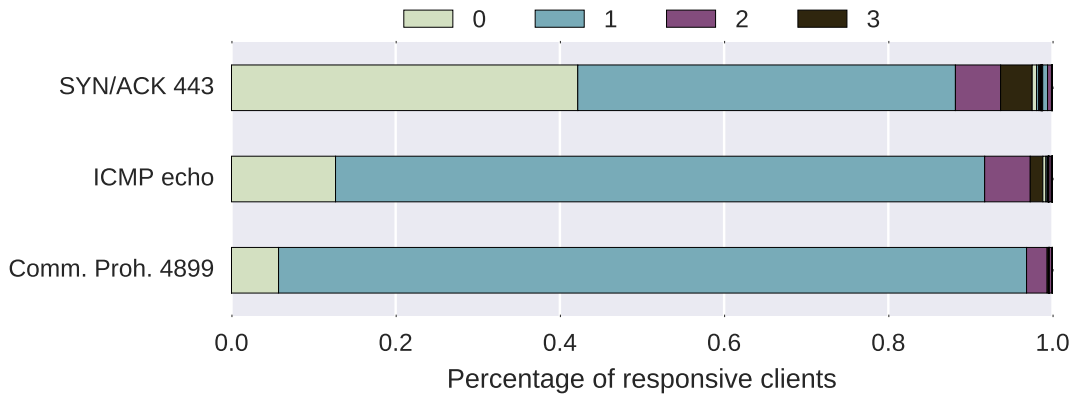


Figure 6.1: Deviation differences for selected probe types

these incoming packets. Initially, we consider all probes to a specific host for our analysis. In cases where the measured TTL values differ between the probe types, we select the minimum value of any test. The intuition of this is straightforward: whenever a firewall or router answers instead of the probed system, the number of hops between them and our probing server is smaller. Hence, by choosing the minimum TTL value, we ensure that we measure the longest path between us and the host responding to the probe. Therefore, if the probed system answers to one probe whereas all others are responded to by the firewall, we still measure the accurate value for the system in question.

The results of applying this methodology on the data set are shown in Table 6.1. We observe that, with respect to the total number of responding systems, 26.1% of the measured TTLs match the ground truth. Moreover, 92.2% of the values are within a threshold of +/-1, and almost 97% within +/-2. In the following, we analyze the results for specific tests in more detail, and discuss potential reasons for the observed deviations.

The deviation between the measured and actual values is shown in Figure 6.1 for ICMP echo, Communication Prohibited to TCP port 4899, and SYN/ACK for TCP port 443. We can observe that for ICMP echo, 12.8% of measured TTLs were correct, whereas an additional 78.8% were off-by-one, *i.e.,* 91.6% of the

Deviation	Amount	Fraction	Cumulated Fraction
+/-0	82,629	26.1%	26.1%
+/-1	208,891	66.1%	92.2%
+/-2	14,623	4.6%	96.9%
+/-3	4,684	1.5%	98.4%
more	5,185	1.6%	100%

Table 6.1: Accuracy of measured TTLs (direct probes only)

measured TTLs were within a threshold of +/-1. For *Communication Prohibited* on port 4899, we observe that 96.8% of the values are within +/-1, whereas 91% are off-by-one. This appears natural to the scenarios we discussed: ICMP echo requests will often be answered by routers and firewalls due to network address translation. Although SYN on TCP port 443 was only responsive on 5.9% of the hosts, the results are quite interesting. We observe that for 42.2% of the hosts which responded to such a probe, the TTL value could be correctly measured. In addition, another 45.9% were off-by-one, resulting in 88% of the values being within a threshold of +/-1. We argue that this is caused by nature of TCP, *i.e.,* we only receive a SYN/ACK in case a service is listening on the probed system. This can either occur if the connection directly reached the probed system, *i.e.,* it is not behind a NAT or the corresponding port is forwarded, or there could be a chance that a public-facing administrative interface is being exposed for service needs [140]. Therefore, it is plausible that such routers may respond to HTTPS requests, explaining the high number of our off-by-one measurements.

Next to probing of the target system itself, we can probe neighboring hosts. More specifically, we probe direct neighbors (IP +/-1) and additionally rely on previous measurements aimed towards other hosts within the same /24 network. In doing so, we find that both types of probing increase the coverage. In our experiment, we found that directly probing neighbors increases the number of measurable TTLs by 69,399, resulting in a total coverage of 73.4%. Taking into account all information from hosts within the same /24 network increases the coverage more drastically (by 168,730 hosts), yielding TTL values for 91.6% of all hosts. At the same time, the accuracy remains similar, with 27% of the probed values matching the ground truth. For +/-1, we can correctly measure the TTL in 88.9% of the cases, and 94.3% of all measurements are within a threshold of +/-2. Given these results for coverage and accuracy, we note that combining different types of probing towards a single host with horizontal probing of the system's neighbors, namely 91.6% for coverage and 94.3% for accuracy, allows us to measure the TTL within a threshold of +/- 2 for 86.4% of all connecting hosts.

6.5.2 Spoofed Traffic

Next to the benign data set, for which we can measure the TTL within a small threshold for the majority of the hosts correctly, we wanted to investigate how well HCF would be suited for spoofed traffic. To that end, we set up a honeypot running a vulnerable version of NTP server prone to becoming an amplifier for DRDoS attacks. To avoid unnecessarily harming the spoofed targets while still pretending to be attractive to adversaries, the outgoing bandwidth was limited, *i.e.,* we answered to at most two monlist requests per host per minute. We did not announce the IP address of this machine in any manner and hence assume that no legitimate traffic would be directed to the host. Instead, incoming NTP

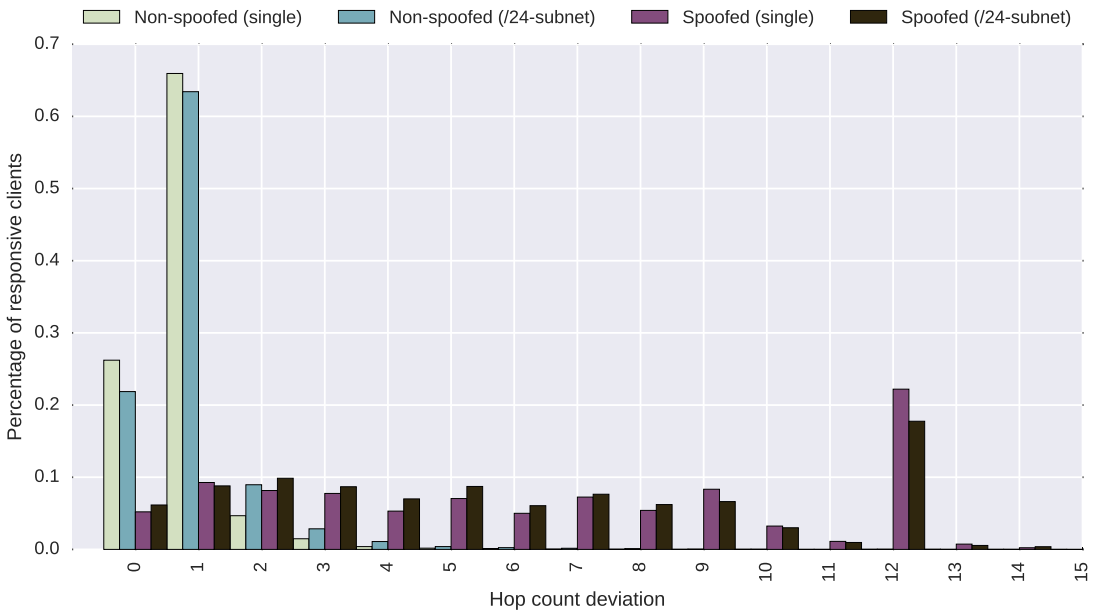


Figure 6.2: Deviation difference between spoofed and non-spoofed traffic

requests are either due to scanning, or spoofed packets sent by an attacker. In a time-period of 96 hours, we recorded 5,616 distinct alleged sender addresses, for which we could gather direct probe results in 3,983 cases (70.9%). This slightly higher coverage (compared to the benign data) can be explained by the fact that most attacks are targeting servers, which also are more likely to expose services we actively probe for.

Before conducting any of our measurements, one property of the spoofed traffic became apparent: more than 99% of all incoming packets had an assumed initial TTL of 255. This specific feature, however, should not be used solely to detect spoofed traffic, since the initial TTL can be changed without much effort by the attacker. Therefore, we normalized the TTL value as outlined before.

Figure 6.2 shows the comparison between the measured TTL values and the TTL values extracted from incoming packets, for both benign and spoofed data sets. While we can clearly observe that for the majority of benign clients, the TTL can be guessed within a threshold of +/-2, we note that no such trend is visible for spoofed traffic.

6.5.3 Implications

In this section, we outlined the results of our experiments on benign and spoofed data sets to evaluate a feasible margin of error for HCF. With respect to those data sets, we find that distinguishing between benign and spoofed traffic appears to yield

useful results when using a threshold of 2. The reasons for the imprecision of the measurements are manifold, *e.g.,* when a client is behind a NAT or incoming traffic to the machine is filtered by a firewall. Therefore, a TTL-based defense mechanism must make a trade-off between false positives and false negatives, respectively. Based on the data sets we analyzed, if a TTL-based defense mechanism was to be deployed to protect a service against becoming an unwilling actor in an attack, over 86.4% of the benign traffic could pass, while more than 3/4 of spoofed packets could be dropped, thus avoiding to harm the targets.

Depending on the type of attacked hosts, this distinction might be even easier to make. Nevertheless, any TTL-based defense relies on one tacit assumption: an attacker cannot learn the correct TTL value for an arbitrary victim and an amplifier of his choosing. Therefore, in the following, we discuss the feasibility of a method in which the attacker can learn the TTL value (within a given threshold).

6.6 Methodology for Estimating Hop Count Value

So far we showed that deploying a TTL-based filtering at the server side would require some tolerance interval to be functional and avoid collateral damage by incorrectly classifying legitimate traffic. In this section, we assess if an attacker can actually bypass the filtering by predicting the correct hop count value between the hosts and properly adjusting the TTL value. That is, we present a methodology for estimating the hop count value between amplifiers and victims.

6.6.1 Key Idea and Attacker Model

Our key idea lies on the observation that paths between arbitrary locations to a selected destination share (small) segments of the path. We leverage the fact that such path information can be learned by an attacker to estimate the number of hops a packet has to traverse from one location to another. To learn subpaths, we (i) probabilistically model known paths obtained via traceroutes, and (ii) combine this knowledge with BGP routing information. Figure 6.3 shows our idea for estimating the distance (number of hops) between an amplifier (M) and a victim (V). For our methodology, we use the common approach for representing the Internet, which is a graph where nodes are the autonomous systems and edges are the peerings (routing links) between them. Additionally, we assign weights to the nodes to denote the hop count number within the individual AS. One way to build such a graph that illustrates the AS-level topology of the Internet is to use available BGP data to discover the connectivity information for the ASes. Nevertheless, studies have shown that BGP data is only available to a limited extent, therefore the Internet AS-level topology is partially hidden [141, 142]. However, our methodology does not primarily rely on the available BGP data, but rather on the traceroute information an attacker can obtain. We use the BGP

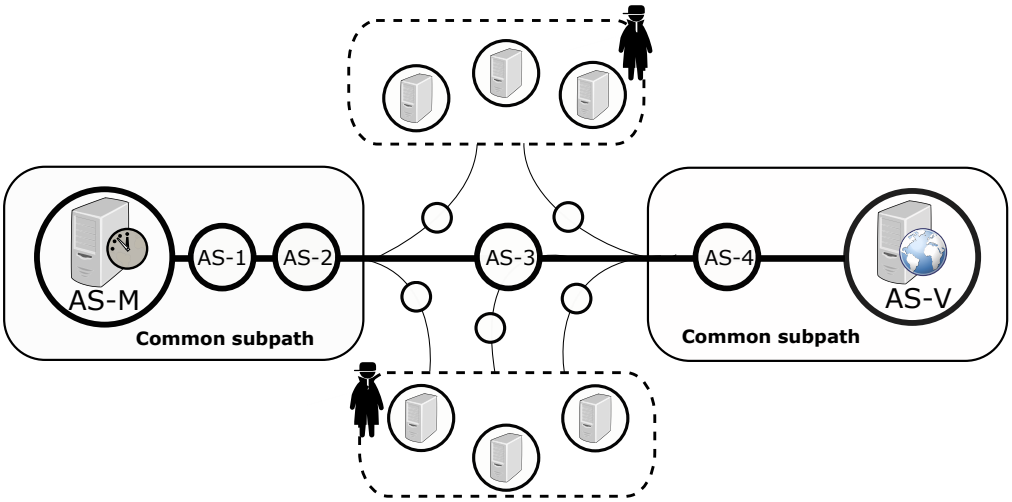


Figure 6.3: Approach to estimate the hops between amplifier (M) and victim (V)

data, when available, as a complement to the traceroute data in order to discover the missing ASes, and to subsequently calculate the number of hops.

Our attacker (A) aims at evading any TTL-based filter or, at least, reduce its effectiveness in mitigating amplification attacks. His main goal is to predict the hop count between the amplifier and the victim as close as possible to the correct one. Then, he can use this value (*i.e.,* when spoofing the victim's TTL) to craft requests which are deemed to be legitimate to the server, *i.e.,* amplifier[1].

In theory, there are few approaches that the attacker may follow to learn the correct TTL value. First, he may learn the TTL value by actively or passively monitoring traffic anywhere on the route, and then probe the destination in order to calculate the remaining part of the route. This approach is neither realistic nor practical because the attacker has to be present at every route R_i between M_i and the victim V. Second, if the attacker can position a probe either in the network of M or V, he can easily measure the TTL value by tracerouting to the other host.

For a more realistic scenario, we restrict the attacker's capabilities. Figure 6.3 illustrates this attacker model. Similar to the reverse traceroute method [143], our attacker is capable of probing from random, distributed locations and can use any publicly available online resources to traceroute to the amplifier and to the victim (*e.g.,* RIPE Atlas[144] or looking glass servers). However, he does not have

[1]When spoofing the TTL, the attacker also has to take into account the hop count between himself and the amplifier. Learning this value is trivial by using some of the methods explained in Section 6.4.1. In case the attacker does not want to expose his IP address to the amplifier, he can use different indirect approaches. For example, he can select as a victim, a host, that he has complete control over and then derive the TTL value from the incoming packet.

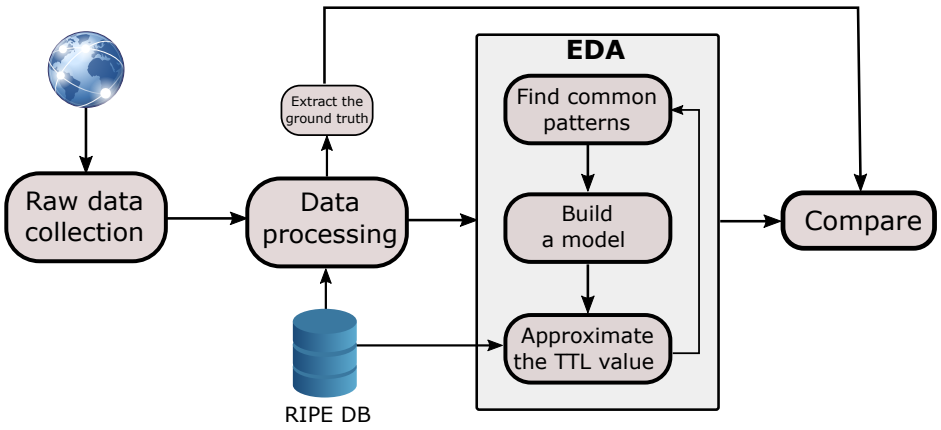


Figure 6.4: Workflow of the methodology

control over the amplifier and not necessarily full control over the probes.

We restrict neither the location of the amplifiers nor the victims, *i.e.,* they can be located at arbitrary network locations. We assume that A can obtain a set of amplifiers (*e.g.,* NTP, DNS), all of which deploy TTL-based filtering and respond to valid requests only[2].

6.6.2 Methodology

We propose a methodology for estimating the distance between hosts on the Internet through an Exploratory Data Analysis (EDA)[3]. Our methodology is comprised of three main components, namely, data collection, data processing, and EDA. Figure 6.4 illustrates the methodology we propose in this work.

6.6.2.1 Data Collection

First, depicted in the data collection component, the attacker collects traceroute data for the victim and the amplifier(s). The attacker launches traceroutes to the targeted locations from a globally distributed set of hosts on the Internet such as RIPE Atlas [144]. Note that the distribution of the selected hosts is required to be global such that there will be a diversity of the paths, allowing us to predict TTLs for arbitrarily chosen victims.

[2]We assume that the amplifiers have deployed HCF to protect against amplification attacks, therefore "valid" protocol requests are those with matching TTL value.

[3]Exploratory Data Analysis is not a method or a technique, but rather a philosophy for data analysis that employs a variety of techniques.

6.6.2.2 Data Processing

Second, in the data processing component, we have to ensure that the relevant data collected in the previous stage is complete and usable. In an ideal world, tracerouting returns a complete path including all the IP addresses and ASes on the way up to the destination. In practice, the collected data from the previous phase is usually imperfect, with a plethora of missing connecting hops [145]. Such data can pose difficulties in effective data analysis; therefore, we need to develop certain methods for efficient data scrubbing. First, we discard all the traceroutes that are missing more than a certain percentage (*e.g.,* 50%) of the intermediate hops. Also, we ignore traceroutes that cannot reach at least the AS of the destination. In the case where the destination address belongs to the same AS as the last replying node, we make an intuitive assumption that this is the last AS in the path, and we supplement the route with the AS number of the last replying node. We then continue filling up the gaps of the unknown ASes due to private IP addresses within the traceroute. Private addressing might occur when a packet passes through someone's internal network with implemented Multiprotocol Label Switching (MPLS) routing [146]. In such cases, we assume that the border AS, the one with a public IP address before the MPLS routing, is the correct one, and we fill in the gaps accordingly. Finally, to fill in the remaining missing hops, we apply a technique that employs the publicly available BGP data. The BGP data assists in the discovering of the neighboring AS [4] and helps us to bridge the gap between two known autonomous systems. Note that this technique can only complete the lacking AS numbers, but not the actual hops (and their IP addresses).

6.6.2.3 Exploratory Data Analysis

Once the data is processed, *i.e.,* prepared for analysis, we dissect the data set using the EDA approach. This stage of the methodology repeats for every victim and it involves three subsequent steps.

Find common patterns. Finding common patterns is the first step in the data exploration. This method transforms the paths from detailed traceroutes with IP addresses of the hops to coarse-grained ones with only AS-level paths and their weights, *i.e.,* the number of hops in each AS for a particular traceroute.

Build a model. This method assists in constructing a probabilistic table that identifies the likelihood of an AS to be part of the route between amplifier and victim. If all collected traceroutes pass through a particular AS, say AS-1, on the

[4] A neighbor (or peering) autonomous system is the one that the AS directly interconnect with in order to exchange traffic.

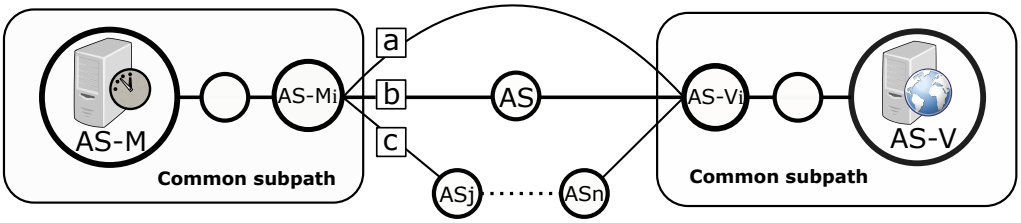


Figure 6.5: Connecting border ASes (AS-Mi and AS-Vi)

way to the target location T, the method denotes the probability of 1 that the AS-1 exist as a hop on the way to T. Moreover, this method also considers the average number of hops within the AS and the distance of the AS from the target. The average number is the AS internal hop count value, and it may vary due to routing-related reasons such as load balancing. To identify the border autonomous systems (in the next step), we need to define the distance as a number of hops that a particular AS is distant from the target AS. For example, the AS the target T belongs to always has a probability of 1 and distance 0.

Approximate the TTL value. The probabilistic modeling helps in building a partial path between two hosts. Consider the scenario illustrated in Figure 6.5. The model identifies with a degree of certainty the common subpaths of the target and the source. Furthermore, it estimates the hop count value of these subpaths. To estimate the final hop count value, we need to bridge these two subpaths with the missing intermediate AS(s). To this end, we apply techniques based on the available BGP data such that the final result is a fully connected AS-level path.

Initially, we identify the border autonomous systems (labeled as AS-Mi and AS-Vi in Figure 6.5), *i.e.,* the last certain (most distant) AS in the common subpaths. With respect to the possible missing hops for connecting these two subpaths, we distinguish three different scenarios (marked with a, b and c in Figure 6.5):

Direct connection (a): When a direct peering between the border autonomous systems exists, *i.e.,* AS-Mi is in the neighborhood[5] of AS-Vi and vice versa, and the intersection set of the AS-Mi and AS-Vi neighbors is empty; we assume that the border ASes are directly connected (AS-Mi $\longleftrightarrow$ AS-Vi).

One-hop connection (b): To identify the single connecting point in between, accordingly, we have to check the neighbors of the border ASes. In the case

[5]Peering ASes are ASes which directly interconnect with each other. We obtain this information from the available BGP data.

where only one intersecting AS exists, we assume that this particular AS is the connecting point. If the intersection set contains more than one common AS, we refer to our probability table. We then accordingly choose the AS with the biggest probability to be a part of the route.

N-hop connection (c): A more complex scenario is when two or more intermediate AS are missing. In such a scenario, we build a tree of possible subpaths by examining additional two levels[6] of neighbors. Upon building up the tree of all possible paths, we test every branch over the database of available BGP routes and the pre-computed table of probabilities. In case the branch is present in the BGP routing database, we deem that particular route to be the accurate one.

Once the bridging subpath is identified, we add up the average hop count of the connecting ASes to the sum of the hop count value estimated for the subpaths.

6.7 Experimental Setup and Results

In the following, we describe the data set used to evaluate our approach. Subsequently, we present and discuss the experimental results of the evaluation.

6.7.1 Data Set

To evaluate the proposed methodology, we mainly use services provided by the RIPE Atlas network [144], which is the largest Internet measurement network built by RIPE NCC. Moreover, they provide an API for creating different types of measurements and for collecting the data in a structured format. In the following, we list the services and data sources used for our experimental evaluation.

1. RIPE Atlas probes: To attain a global coverage and also to have a possibility to obtain the ground truth, we use the RIPE Atlas network of probes [144] as a basis for our experiments. We observe that this network has around 9,000 active probes, spread across 181 countries and 3,386 ASes [147]. Such a global coverage fulfills the requirements for our experimental evaluation. Moreover, the platform give us the flexibility for requesting custom measurements, in our case traceroutes, by selecting any of the deployed active probes. This flexibility is of particular importance for our experiments since we can select a subset of nodes with different geographical and logical locations to collect the traceroute data. Additionally, when a probe acts as a victim in our leave-one-out analysis (which we outline in the following), we can easily

[6]Statistics [144] show that average length of AS-level paths is four, therefore we bound the subpath examination to two levels, *i.e.,* we can examine paths of at least six hops.

obtain the ground truth by running traceroute measurement from the probes to the amplifiers.

2. BGP data: When the collected traceroute data is not enough for making the final assessment of the connectivity between the ASes, we utilize available BGP data. In order to infer the AS-level connectivity, we use RIPE Atlas as an accurate source for BGP data. Also the BGP data helps to obtain a ground truth of individual ASes.

3. Amplifiers: To investigate the real-world implications of our attack, we scanned for chargen amplifiers on the Internet. In total, we randomly selected 16 such servers.

6.7.2 Leave-one-out Evaluation

To evaluate the performance of our methodology, we use a leave-one-out (*L-1-O*) evaluation approach, in which every probe acts like a victim at a selected time. Informally, for a data set with P probes, we perform P experiments with $P-1$ training cases and one test case. In other words, for every experiment we temporarily remove one probe from the data set and select that particular probe as our victim. Upon fixing the probe P_i as a victim V, the model is rebuilt upon this newly defined set.

Suppose that $P = p_1, \ldots, p_n$ is a set of probes, $M = m_1, \ldots, m_l$ set of amplifiers, and $R = r_{11}, \ldots, r_{nm}$ set of traceroutes where r_{ij} is a traceroute from p_i to m_j. For ease of exposition, we use the notation $p_i \Rightarrow_R M$ to describe a set of all traceroutes from p_i to every member of the set M. Applying the L-1-0 approach to the methodology works as follows:

1. Collect the traceroute data ($R \bigcup \{p_i \Rightarrow_R P \setminus \{p_i\} | i = 1, \ldots, n\}$).

2. Process the data and extract the ground truth.

3. Remove probe p_i from P ($P \setminus \{p_i\}$) and set $V = p_i$, where V is the victim.

4. Extract the ground truth for p_i to M *i.e.,* the distance from $p_i \Rightarrow_R M$.

5. Run the EDA using the remaining data.

6. Repeat step 3-5 for $i = 1, \ldots, n$

L-1-O in practice. We apply the L-1-O method on a set of 40 random RIPE Atlas probes, located in different ASes, and 16 randomly distributed chargen amplifiers. We first collect the required data, namely, we obtain the path from every probe to all of the 16 amplifiers, and also between the probes within the set. We use the RIPE Atlas REST API to create IPv4 traceroutes using ICMP packets

	Amount	Fraction	Cumulated Fraction
+/-0	78	13.2%	13.2%
+/-1	170	28.7%	28.5%
+/-2	132	22.3%	56.3%
+/-3	49	8.3%	69.1%
more	164	27.7%	100%

Table 6.2: Overall performance of the methodology

and hops limit of 32. In order to get more precise paths and avoid measurements inconsistencies caused by load balancing routers, we employ the paris traceroute measurement tool [148].

Once the traceroute data is collected and the data set is processed, *i.e.,* cleaned up using the method described in Section 6.6.2.2, we pass the data through step 3 to 6 from the L-1-0 approach. In such experimental setup, L-1-O theoretically can evaluate 640 TTL predictions, *i.e.,* paths from 16 amplifiers to 40 victims. Unfortunately, because of the incompleteness of the traceroute data as well as instability of some of the probes, the method was able to predict and evaluate around 593 (92.6%) individual paths.

6.7.3 Overall Performance

Table 6.2 shows the overall performance of our methodology. The experimental results show that using our methodology, an attacker can predict correctly without any deviation roughly 13% of the paths between the amplifiers and the victims, *i.e.,* 13% of the measured hop counts match the ground truth. However, we showed in Section 6.5 that, with a tolerance of +/-2, a TTL-based defense could block over 75% of spoofed traffic, while allowing 86.4% of benign traffic to pass. Therefore, when we take this threshold into consideration, our methodology is effective for 56.3% of the paths.

Moreover, we observe that applying our methodology to a set of randomly chosen amplifiers, the attacker can isolate amplifiers for which he can predict the hop count value between the amplifier and any arbitrary victim with higher accuracy. Thus, he can bypass the TTL-based defense running on the amplifier and exploit it for a DRDoS attack. Figure 6.6 illustrates the average hop count deviation per amplifier and shows that the attacker can, indeed, sample a set of *good* amplifiers. We see several explanations for such a deviation among the amplifiers. The geographical and logical location of the amplifiers and the victims plays an important role. As we discussed before, the limitation of the BGP data makes our methodology not equally precise for all the AS. Also another cause is the inconsistency of the collected data between BGP data and traceroute path caused by Internet Exchange Points and sibling ASes managed by the same institution. However, these results show that even with a low threshold value at

the amplifier, by wisely choosing amplifiers to use, an attacker is able to circumvent any TTL-based defense against DRDoS attacks.

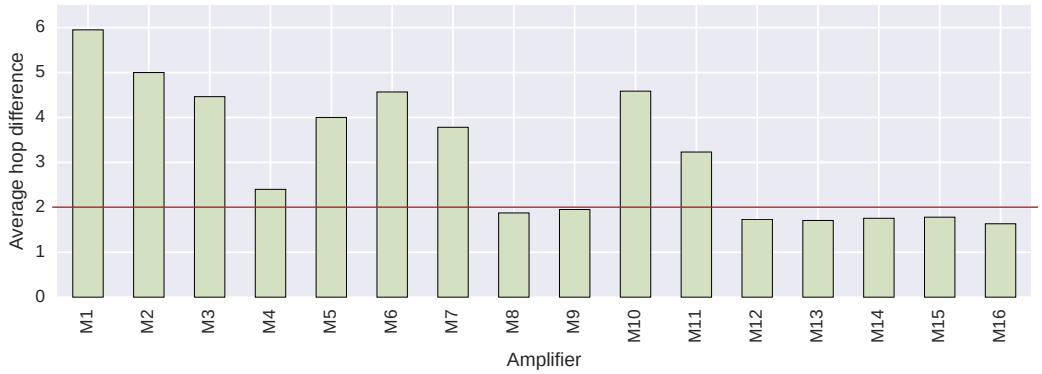


Figure 6.6: Average hop deviation per amplifier

6.8 Conclusion

In this chapter, we evaluated the feasibility of using Hop Count Filtering to mitigate DRDoS attacks. To that end, we detailed how a server can use active probing to learn TTLs of alleged packet senders. Based on data sets of benign and spoofed NTP requests, we find that with a tolerance of +/-2, a TTL-based defense could block over 75% of spoofed traffic, while allowing 86.4% of benign traffic to pass. Subsequently, however, we show that an attacker can use a combination of tracerouting and BGP data to build statistical models, which allows him to estimate the TTL for his target within that tolerance level. Hence, by wisely choosing which amplifiers to use, he is able to circumvent any TTL-based defense against DRDoS attacks. We therefore argue that any (current or future) defensive system based on TTL values can be bypassed in a similar fashion, and find that future research must be steered towards more fundamental solutions to thwart any kind of IP spoofing attacks.

7
Conclusion

This dissertation presented a line of work that tackles accountability from few different perspectives. In particular, we make the following fourfold contributions.

Our first contribution is the design of BackRef, an accountability mechanism for ACNs, in a form of a repudiation technique for the proxy nodes. To the best of our knowledge, the proposed mechanism is the first that is applicable to low-latency networks and does not require group managers and credential issuers. BackRef is provably secure, requires little overhead, and can be adapted to a wide range of anonymity systems. Our mechanism is based on the newly introduced concept of pseudonymous signatures that may be of independent interest. We formally defined the important properties of the proposed mechanism. In particular, we formalized anonymity and no forward traceability as observational equivalence relations, and backward traceability and no false accusation as trace properties. We conducted a formal security analysis using automated cryptographic protocol verifier ProVerif, establishing the aforementioned security and privacy properties against a strong adversarial model. We strongly believe that both the definitions and the security analyses are of independent interest since they are the first for the onion routing protocol. We analyzed and discussed important systems issues such as white-listing, log storage, non-cooperating nodes, and the last mile problem, that any reactive accountable anonymous communication network might encounter, and presented plausible options towards deploying BackRef in practice.

Our second contribution is the design of Oblivion, a universal framework for providing the foundation to support the enforcement of the European *right to be forgotten* legislation in a scalable and automated manner. The framework enables a user to automatically identify personal information in a given article and the indexing system to automatically verify the user's eligibility. We formally defined the protocol in the applied π-calculus and automatically verify the censorship resistance property using ProVerif. Finally, we conducted comprehensive evaluations of Oblivion on existing articles, showing that the framework incurs only minimal overhead.

The third contribution of the thesis is a design of a framework for modeling and reasoning on global-scale threats. We presented a model of the Internet infrastructures based on property graphs. To mind the data from our model and to quantify the results, we presented a combination of taint-style techniques and propagation rules, which is automatically translated in graph traversals. We assessed our approach on 1.8M data items acquired from the top 100K Alexa domains. For selecting attacker and victim candidates, we presented six metrics. Then, we measured the impact of three different attack scenarios, which are based on the Great Cannon attack, the PRISM program, and the DDoS against Dyn.com. Our results showed that already just a few players may have an extensive power. More precisely, 14 countries and 14 autonomous systems can, directly or indirectly, affect the security of about 23% of the considered websites. Finally, our results

indicated that the recent DDoS attack against Dyn.com was the result of a careful choice.

With our final contribution, we showed that a defensive mechanism for DRDoS attacks based on TTL values can be easily bypassed by approximating the TTL value. First, we analyzed several ways of probing for the TTL of an alleged sender, using different types of probes towards a host in question including horizontal probing of its neighbors. We showed that this process is prone to errors and frequently tedious in practice, raising the need for a certain tolerance in TTL-based defenses. More precisely, we showed that an error margin of +/-2 must be allowed to enable at least 86.4% of benign traffic to pass, while dropping more than 75% of spoofed traffic. Then, taking into consideration the error margin, we showed that an attacker can use a combination of tracerouting and BGP data to build statistical models, which allows him to estimate the TTL for his target within that tolerance level. Moreover, we showed that by wisely choosing amplifiers to use, the attacker is able to circumvent any TTL-based defense against DRDoS attacks. Finally, we argued that any (current or future) defensive system based on TTL values can be circumvented in a similar fashion, and find that future research must be steered towards more fundamental solutions to thwart any kind of IP spoofing attacks.

Bibliography

Author's Papers

[1] Michael Backes, Jeremy Clark, Aniket Kate, Milivoj Simeonovski, and Peter Druschel. "BackRef: Accountability in Anonymous Communication Networks." In: *Proceedings of the 12th International Conference on Applied Cryptography and Network Security (ACNS 2014)*. 2014.

[2] Milivoj Simeonovski, Fabian Bendun, Muhammad Rizwan Asghar, Michael Backes, Ninja Marnau, and Peter Druschel. "Oblivion: Mitigating Privacy Leaks by Controlling the Discoverability of Online Information." In: *Proceedings of the 13th International Conference on Applied Cryptography and Network Security (ACNS 2015)*. 2015.

[3] Milivoj Simeonovski, Giancarlo Pellegrino, Christian Rossow, and Michael Backes. "Who Controls the Internet?: Analyzing Global Threats using Property Graph Traversals." In: *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*. ACM.

[4] Michael Backes, Thorsten Holz, Christian Rossow, Teemu Rytilahti, Milivoj Simeonovski, and Ben Stock. "On the Feasibility of TTL-Based Filtering for DRDoS Mitigation." In: *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2016)*. Springer, 2016.

[5] Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. "A Survey on Routing in Anonymous Communication Protocols." In: *ACM Computing Surveys (CSUR)* 51.3 (2018), 51:1–51:39.

[6] Muhammad Rizwan Asghar, Michael Backes, and Milivoj Simeonovski. "PRIMA: Privacy-Preserving Identity and Access Management at Internet-Scale." In: *Proceedings of the IEEE International Conference on Communications (ICC 2018)*. IEEE, 2018.

[7] Patrick Speicher, Marcel Steinmetz, Robert Künnemann, Milivoj Simeonovski, Giancarlo Pellegrino, Joerg Hoffmann, and Michael Backes. "Formally Reasoning about the Cost and Efficacy of Securing the Email Infrastructure." In: *Proceedings of the 3nd IEEE European Symposium on Security and Privacy (Euro S&P 2018)*. IEEE, 2018.

[8] Milivoj Simeonovski. "POSTER: Quasi-ID: In Fact, I Am a Human." In: *Proceedings of the 21st ACM Conference on Computer and Communication Security (CCS 2014)*. ACM, 2014.

Other references

[9] *The New Threat: Targeted Internet Traffic Misdirection*. Online: `http://research.dyn.com/2013/11/mitm-internet-hijacking/`. 2013.

[10] *UK traffic diverted through Ukraine*. Online: `http://research.dyn.com/2015/03/uk-traffic-diverted-ukraine/`. 2015.

[11] Gabi Nakibly, Jaime Schcolnik, and Yossi Rubin. "Website-Targeted False Content Injection by Network Operators." In: *Proceedings of the 25th USENIX Security Symposium (SEC 2016)*. USENIX Association, 2016.

[12] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, and Vern Paxson. "An Analysis of China's Great Cannon." In: *Proceedings of the 5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 15)*. USENIX Association, 2015.

[13] David Chaum. "The dining cryptographers problem: Unconditional sender and recipient untraceability." In: *Journal of Cryptology* 1.1 (1988), pp. 65–75.

[14] Henry Corrigan-Gibbs and Bryan Ford. "Dissent: accountable anonymous group messaging." In: *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010)*. ACM, 2010.

[15] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. "Anonymous Connections and Onion Routing." In: *Proceedings of the 18th IEEE Symposium on Security and Privacy (S&P 1997)*. IEEE, 1997.

[16] David Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." In: *Communications of the ACM (CACM)* 24.2 (1981), pp. 84–90.

[17] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. *Mixmaster Protocol—Version 2*. IETF Internet Draft. Online: http://mixmaster.sourceforge.net/. 2003.

[18] G. Danezis, R. Dingledine, and N. Mathewson. "Mixminion: design of a type III anonymous remailer protocol." In: *Proceedings of the 24th IEEE Symposium on Security and Privacy (S&P 2003)*. IEEE, 2003.

[19] Ceki Gülcü and Gene Tsudik. "Mixing Email with Babel." In: *Proceeding of the 3rd Symposium on Network and Distributed System Security (SNDSS 1996)*. The Internet Society, 1996.

[20] Prateek Mittal and Nikita Borisov. "ShadowWalker: peer-to-peer anonymous communication using redundant structured topologies." In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS 2009)*. ACM, 2009.

[21] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. "AP3: cooperative, decentralized anonymous communication." In: *Proceedings of the 11st ACM SIGOPS European Workshop*. ACM, 2004.

[22] Alexander W. Janssen. *Tor madness reloaded*. Online: `http://itnomad.wordpress.com/2007/09/16/tor-madness-reloaded/`. 2007.

[23] William Weber. *Raided for operating a Tor exit node*. Online: `http://raided4tor.cryto.net/`.

[24] Alexander W. Janssen. *The Onion Router: A brief inftroduction and legal aspects*. Online: `http://yalla.ynfonatic.de/media/lbw2007/tor_talk-LBW2007.pdf`. 2007.

[25] John Zorabedian. *Couple hosting Tor exit node raided by cops investigating child abuse*. Online: `https://nakedsecurity.sophos.com/2016/04/07/couple-hosting-tor-exit-node-raided-by-cops-investigating-child-abuse/`. 2016.

[26] Stefan Köpsell, Rolf Wendolsky, and Hannes Federrath. "Revocable Anonymity." In: *Proceedings of the International Conference on Emerging Trends in Information and Communication Security (ETRICS 2006)*. Springer, 2006.

[27] Luis Von Ahn, Andrew Bortz, Nicholas J. Hopper, and Kevin O'Neill. "Selectively Traceable Anonymity." In: *Proceedings of the 6th Workshop on Privacy Enhancing Technologies (PET 2006)*. Springer, 2006.

[28] C. Diaz and B. Preneel. "Accountable anonymous communication." In: *Security, Privacy, and Trust in Modern Data Management*. Springer, 2007.

[29] Philippe Golle. "Reputable mix networks." In: *Proceedings of the 4th Workshop on Privacy Enhancing Technologies (PET 2004)*. Springer, 2004.

[30] Jeremy Clark, Philippe Gauvin, and Carlisle Adams. "Exit Node Repudiation for Anonymity Networks." In: *On the Identity Trail: Privacy, Anonymity and Identity in a Networked Society*. Oxford University Press, 2009.

[31] Peter C. Johnson, Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. "Nymble: Anonymous IP-Address Blocking." In: *Proceedings of the 7th Workshop on Privacy Enhancing Technologies (PET 2007)*. Springer, 2007.

[32] Ryan Henry and Ian Goldberg. "Formalizing Anonymous Blacklisting Systems." In: *Proceedings of the 32nd IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2011.

[33] Ian Goldberg, David Wagner, and Eric Brewer. "Privacy-enhancing technologies for the Internet." In: *Proceedings of the IEEE Compcon*. IEEE, 1997.

[34] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: the second-generation onion router." In: *Proceedings of the 13th USENIX Security Symposium (SEC 2004)*. USENIX Association, 2004.

[35] Michael K. Reiter and Aviel D. Rubin. "Crowds: anonymity for Web transactions." In: *ACM Transactions on Information and System Security (TISSEC)* 1.1 (1998), pp. 66–92.

[36] Ian Goldberg and Adam Shostack. *Freedom Network 1.0 Architecture and Protocols*. Tech. rep. Zero-Knowledge Systems, 1999.

[37] Aniket Kate, Greg M. Zaverucha, and Ian Goldberg. "Pairing-Based Onion Routing with Improved Forward Secrecy." In: *ACM Transactions on Information and System Security (TISSEC)* 13.4 (2010), 29:1–29:32.

[38] George Danezis and Ian Goldberg. "Sphinx: A Compact and Provably Secure Mix Format." In: *Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P 2009)*. IEEE, 2009.

[39] Oliver Berthold, Hannes Federrath, and Stefan Kopsell. "Web MIXes: A System for Anonymous and Unobservable Internet Access." In: *Proceedings of the International workshop on Designing privacy enhancing technologies (PETS 2000)*. Springer, 2000.

[40] Michael J. Freedman and Robert Morris. "Tarzan: a peer-to-peer anonymizing network layer." In: *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*. ACM, 2002.

[41] Andreas Pfitzmann and Marit Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. Online: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf. 2010.

[42] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. "Dissent in numbers: making strong anonymity scale." In: *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2012)*. USENIX Association, 2012.

[43] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. "Proactively Accountable Anonymous Messaging in Verdict." In: *Proceedings of the 22nd USENIX Security Symposium (SEC 2013)*. USENIX Association, 2013.

[44] George Danezis and Len Sassaman. "How to Bypass Two Anonymity Revocation Schemes." In: *Proceedings of the 8th Symposium on Privacy Enhancing Technologies (PETS 2008) (PETS 2008)*. Springer, 2008.

[45] *Exonerator: Archive for relays running in the past*. Online: `https://metrics.torproject.org/exonerator.html`.

[46] Kevin S. Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas C. Sicker. "Low-resource routing attacks against tor." In: *Proceedings of the 6th ACM Workshop on Privacy in the Electronic Society, (WPES 2007)*. ACM, 2007.

[47] Andrei Serjantov and Peter Sewell. "Passive Attack Analysis for Connection-Based Anonymity Systems." In: *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS 2003)*. 2003.

[48] Lasse Øverlier and Paul F. Syverson. "Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services." In: *Proceedings of the 7th Workshop on Privacy Enhancing Technologies (PET 2007)*. Springer, 2007.

[49] Aniket Kate and Ian Goldberg. "Using sphinx to improve onion routing circuit construction." In: *Proceedings of the 14th International Conference on Financial Cryptography and Data Security (FC 2010)*. Springer, 2010.

[50] Michael Backes, Aniket Kate, and Esfandiar Mohammadi. "Ace: an efficient key-exchange protocol for onion routing." In: *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society (WPES) 2012)*. ACM, 2012.

[51] Dario Catalano, Dario Fiore, and Rosario Gennaro. "Certificateless onion routing." In: *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*. ACM, 2009.

[52] I. Goldberg, D. Stebila, and B. Ustaoglu. "Anonymity and one-way authentication in key exchange protocols." In: *Designs, Codes and Cryptography* 67.2 (2013), pp. 245–269.

[53] J. Camenisch and A. Lysyanskaya. "A Formal Treatment of Onion Routing." In: *Proceedings of the 25th Annual International Cryptology Conference (CRYPTO 2005)*. Springer, 2005.

[54] George Danezis, Claudia Díaz, Carmela Troncoso, and Ben Laurie. "Drac: An Architecture for Anonymous Low-Volume Communications." In: *Proceedings of the 10th International Symposium on Privacy Enhancing Technologies (PETS 2010)*. Springer, 2010.

[55] Erik Shimshock, Matt Staats, and Nicholas Hopper. "Breaking and Provably Fixing Minx." In: *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS 2008)*. Springer, 2008.

[56] 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 (1988), pp. 281–308.

[57] Andreas Haeberlen, Pedro Fonseca, Rodrigo Rodrigues, and Peter Druschel. *Fighting Cybercrime with Packet Attestation*. Tech. rep. `http://www.mpi-sws.org/tr/2011-002.pdf`. MPI-SWS, 2011.

[58] Roger Dingledine and Nick Mathewson. *Tor Protocol Specification*. Online: `https://gitweb.torproject.org/torspec.git/tree/HEAD`. 2008.

[59] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing." In: *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2001)*. Springer, 2001.

[60] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-Speed High-Security Signatures." In: *Proceedings of the 13th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2011)*. Springer, 2011.

[61] Michael Backes, Ian Goldberg, Aniket Kate, and Esfandiar Mohammadi. "Provably Secure and Practical Onion Routing." In: *Proceedings of the 25th IEEE Symposium on Computer Security Foundations (CSF 2012)*. IEEE, 2012.

[62] Martín Abadi and Cédric Fournet. "Mobile values, new names, and secure communication." In: *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL 2001)*. ACM, 2001.

[63] Ran Canetti. "Universally composable security: A new paradigm for cryptographic protocols." In: *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001)*. IEEE, 2001.

[64] Bruno Blanchet. "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules." In: *Proceedings of the 14th IEEE Workshop on Computer Security Foundations (CSFW 2001)*. IEEE, 2001.

[65] BackRef. *Introducing Accountability to Anonymity Networks (extended version)*. Online: `http://crypsys.mmci.uni-saarland.de/projects/BackRef/`.

[66] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels. *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005.

[67] Court of Justice of the European Union. *JUDGMENT OF THE COURT (Grand Chamber)*. Online: `http://curia.europa.eu/juris/celex.jsf?celex=62012CJ0131`. 2014.

[68] Google. *Google Transparency Report: Search removals under European privacy law*. Online: `https://transparencyreport.google.com/eu-privacy/overview`.

[69] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-key Cryptosystems." In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[70] Christopher Olston and Marc Najork. "Web Crawling." In: *Foundations and Trends in Information Retrieval* 4.3 (2010), pp. 175–246.

[71] *The Web Robots Pages*. Online: `http://www.robotstxt.org/`.

[72] Michael Backes, Sebastian Gerling, Stefan Lorenz, and Stephan Lukas. "X-pire 2.0 - A User-Controlled Expiration Date and Copy Protection Mechanism." In: *Proceedings of the 29th ACM Symposium on Applied Computing (SAC 2014)*. ACM, 2014.

[73] Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy. "Vanish: Increasing Data Privacy with Self-Destructing Data." In: *Proceedings of the 18th USENIX Security Symposium (SEC 2009)*. USENIX Association, 2009.

[74] Sirke Reimann and Markus Dürmuth. "Timed revocation of user data: Long expiration times from existing infrastructure." In: *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society (WPES 2012)*. ACM, 2012.

[75] Radia Perlman. "The Ephemerizer: Making Data Disappear." In: *Journal of Information System Security* 1 (2005), pp. 51–68.

[76] C. Castelluccia, E. De Cristofaro, A. Francillon, and M.-A. Kaafar. "EphPub: Toward robust Ephemeral Publishing." In: *Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP 2011)*. IEEE, 2011.

[77] SrijithK. Nair, MohammadT. Dashti, Bruno Crispo, and AndrewS. Tanenbaum. "A Hybrid PKI-IBC Based Ephemerizer System." In: *Proceedings of the 22nd International Information Security Conference*. Springer, 2007.

[78] Marco Casassa Mont, Siani Pearson, and Pete Bramhall. "Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services." In: *Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA 2003)*. IEEE, 2003.

[79] S. Pearson and Marco Casassa Mont. “Sticky Policies: An Approach for Managing Privacy across Multiple Parties.” In: *Computer* 44.9 (2011), pp. 60–68.

[80] Sruthi Bandhakavi, Charles C. Zhang, and Marianne Winslett. “Super-sticky and declassifiable release policies for flexible information dissemination control.” In: *Proceedings of the 5th ACM Workshop on Privacy in the Electronic Society (WPES 2006)*. ACM, 2006.

[81] David W. Chadwick and Stijn F. Lievens. “Enforcing "sticky" security policies throughout a distributed application.” In: *Proceedings of the Workshop on Middleware Security (MidSec 2008)*. ACM, 2008.

[82] Google. *Personal Information Removal Request Form (European Data Protection law)*. Online: `https://support.google.com/legal/contact/lr_eudpa?product=websearch`.

[83] Microsoft. *Request to Block Bing Search Results In Europe*. Online: `https://www.bing.com/webmaster/tools/eu-privacy-request`.

[84] Yahoo. *Requests to Block search results in Yahoo Search: Resource for European Residents*. Online: `http://bit.ly/185Lije`.

[85] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.1*. RFC 4346 (Proposed Standard). 2006. URL: `http://www.ietf.org/rfc/rfc4346.txt`.

[86] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. “Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling.” In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics, 2005.

[87] Natural Language Toolkit. *WordNet Interface*. Online: `http://www.nltk.org/howto/wordnet.html`.

[88] Federal Ministry of the Interior. *German National Identity Card*. Online: `http://www.personalausweisportal.de/EN/Citizens/The-New-Identity-Card/`.

[89] The Council of the European Union. *Residence permits for third-country nationals*. Online: `http://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32008R0380`. 2008.

[90] European Parliament, Council of the European Union. *Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC*. Online: `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014R0910`. 2014.

[91] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. "Robust Disambiguation of Named Entities in Text." In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. ACL, 2011.

[92] *Stanford Named Entity Recognizer (NER)*. Online: `http://nlp.stanford.edu/software/CRF-NER.shtml`.

[93] OpenCV. *Open Source Computer Vision*. Online: `http://opencv.org/`.

[94] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. "DBpedia: A Nucleus for a Web of Open Data." In: *Proceedings of the 6th International Semantic Web Conference (ISWC 2007)*. Springer, 2007.

[95] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: A Core of Semantic Knowledge." In: *Proceedings of the 16th International Conference on World Wide Web (WWW 2007)*. ACM, 2007.

[96] *Open Source Web Crawler for Java*. Online: `https://code.google.com/p/crawler4j/`.

[97] Michael Backes, Fabian Bendun, Joerg Hoffman, and Ninja Marnau. "PriCL: Creating a Precedent, a Framework for Reasoning about Privacy Case Law." In: *Proceedings of the 4th International Conference on Principles of Security and Trust (POST 2015)*. Springer, 2015.

[98] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification." In: *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)*. IEEE, 2014.

[99] Susan Landau. "Making Sense from Snowden: What's Significant in the NSA Surveillance Revelations." In: *IEEE Security & Privacy* 11.4 (2013), pp. 54–63.

[100] Jesse Newland. *Large Scale DDoS Attack on github.com*. https://github.com/blog/1981-large-scale-ddos-attack-on-github-com. Blog. 2015.

[101] Réka Albert, Hawoong Jeong, and Albert-László Barabási. "Error and attack tolerance of complex networks." In: *Nature* 406.6794 (2000), pp. 378–382.

[102] Jichang Zhao, Junjie Wu, Mingming Chen, Zhiwen Fang, Xu Zhang, and Ke Xu. "K-core-based attack to the internet: Is it more malicious than degree-based attack?" In: *World Wide Web* 18.3 (2015), pp. 749–766.

[103] David Wheeler and Gregory Larsen. *Techniques for cyber attack attribution*. Tech. rep. Defense Technical Information Center (DTIC), 2003.

[104] Arman Noroozian, Maciej Korczynski, Carlos Hernandez Gañán, Daisuke Makita, Katsunari Yoshioka, and Michel van Eeten. "Who Gets the Boot? Analyzing Victimization by DDoS-as-a-Service." In: *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions, and Defenses(RAID 2016)*. Springer, 2016.

[105] W. Jiang, D. Lee, and S. Hu. "Large-Scale Longitudinal Analysis of SOAP-Based and RESTful Web Services." In: *Proceedings of the 19th IEEE International Conference on Web Services (ICWS)*. IEEE, 2012.

[106] Hyunyoung Kil, Seog-Chan Oh, Ergin Elmacioglu, Wonhong Nam, and Dongwon Lee. "Graph Theoretic Topological Analysis of Web Service Networks." In: *World Wide Web* 12.3 (2009), pp. 321–343.

[107] Sylvain Frey, Yehia Elkhatib, Awais Rashid, Karolina Follis, John Vidler, Nicholas J. P. Race, and Christopher Edwards. "It Bends But Would It Break? Topological Analysis of BGP Infrastructures in Europe." In: *Proceedings of the 1st European Symposium on Security and Privacy (EuroS&P 2016)*. IEEE, 2016.

[108] Kevin RB Butler, Toni R Farley, Patrick McDaniel, and Jennifer Rexford. "A Survey of BGP Security Issues and Solutions." In: *Proceedings of the IEEE* 98.1 (2010), pp. 100–122.

[109] Erik Hjelmvik. *China's Man-on-the-Side Attack on GitHub*. http://bit.ly/2kx4zAE. Blog. 2015.

[110] Giancarlo Pellegrino, Christian Rossow, Fabrice J. Ryba, Thomas C. Schmidt, and Matthias Wählisch. "Cashing Out the Great Cannon? On Browser-Based DDoS Attacks and Economics." In: *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT 15)*. USENIX Association, 2015.

[111] *DDoS Attack Against Dyn (Analysis Summary)*. Online: `https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/`.

[112] RIPE NCC. *RIPE Stat: Information about specific IP addresses and prefixes*. Online: `https://stat.ripe.net/`.

[113] *MaxMind: IP Geolocation and Online Fraud Prevention*. Online: `http://dev.maxmind.com/`.

[114] L. Daigle. *WHOIS Protocol Specification*. RFC 3912 (Draft Standard). Internet Engineering Task Force, Sept. 2004. URL: `http://www.ietf.org/rfc/rfc3912.txt`.

[115] Suqi Liu, Ian Foster, Stefan Savage, Geoffrey M. Voelker, and Lawrence K. Saul. "Who is .Com?: Learning to Parse WHOIS Records." In: *Proceedings of the 2015 ACM Conference on Internet Measurement Conference (IMC 2015)*. ACM, 2015.

[116] Giancarlo Pellegrino, Constantin Tschürtz, Eric Bodden, and Christian Rossow. "jÄk: Using Dynamic Analysis to Crawl and Test Modern Web Applications." In: *Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID 2015)*. Springer, 2015.

[117] Reuven Cohen, Keren Erez, Daniel B. Avraham, and Shlomo Havlin. "Breakdown of the Internet under Intentional Attack." In: *Physical Review Letters* 86.16 (Apr. 2001), pp. 3682–3685.

[118] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. "You Are What You Include: Large-scale Evaluation of Remote Javascript Inclusions." In: *Proceedings of the 19th ACM Conference on Computer and Communication Security (CCS 2012)*. ACM, 2012.

[119] Frank Cangialosi, Taejoong Chung, David R. Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. "Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem." In: *Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS 2016)*. ACM, 2016.

[120] Wenke Lee and Salvatore J. Stolfo. "Data Mining Approaches for Intrusion Detection." In: *Proceedings of the 7th Conference on USENIX Security Symposium (SEC 1998)*. USENIX Association, 1998.

[121] Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. "Modeling and Discovering Vulnerabilities with Code Property Graphs." In: *Proceedings of the 35th IEEE Symposium on Security and Privacy (S&P 2014)*. IEEE, 2014.

[122] Michael Backes, Konrad Rieck, Malte Skoruppa, Ben Stock, and Fabian Yamaguchi. "Efficient and Flexible Discovery of PHP Application Vulnerabilities." In: *Proceedings of the 2nd IEEE European Symposium on Security and Privacy (Euro S&P 2017)*. IEEE, 2017.

[123] Arun Natarajan, Peng Ning, Yao Liu, Sushil Jajodia, and Steve E. Hutchinson. "NSDMiner: Automated discovery of Network Service Dependencies." In: *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM 2012)*. IEEE, 2012.

[124] Ali Zand, Giovanni Vigna, Richard A. Kemmerer, and Christopher Kruegel. "Rippler: Delay injection for service dependency detection." In: *Proceedings of the 33st Annual IEEE International Conference on Computer Communications (INFOCOM 2014)*. IEEE, 2014.

[125] *Technical Details Behind a 400Gbps NTP Amplification DDoS Attack*. Online: `https://goo.gl/j7zWEp`.

[126] Christian Rossow. "Amplification Hell: Revisiting Network Protocols for DDoS Abuse." In: *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS 2014)*. The Internet Society, 2014.

[127] Cheng Jin, Haining Wang, and Kang G Shin. "Hop-count filtering: an effective defense against spoofed DDoS traffic." In: *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003)*. ACM, 2003.

[128] J. Postel. *Internet Protocol*. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864. Internet Engineering Task Force, Sept. 1981. URL: `http://www.ietf.org/rfc/rfc791.txt`.

[129] *Default TTL Values in TCP/IP*. Online: `http://www.map.meteoswiss.ch/map-doc/ftp-probleme.htm`.

[130] Fabrice J Ryba, Matthew Orlinski, Matthias Wählisch, Christian Rossow, and Thomas C Schmidt. "Amplification and DRDoS Attack Defense–A Survey and New Perspectives." In: *arXiv preprint arXiv:1505.07892* (2015).

[131] Jelena Mirkovic and Peter Reiher. "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms." In: *ACM SIGCOMM Computer Communication Review* 34.2 (2004), pp. 39–53.

[132] Stephen M. Specht and Ruby B. Lee. "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures." In: *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems (PDCS 2004)*. ISCA, 2004.

[133] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. "Exit from Hell? Reducing the Impact of Amplification DDoS Attacks." In: *Proceedings of the 23rd USENIX Security Symposium (SEC 2014)*. USENIX Association, 2014.

[134] Vern Paxson. "An analysis of using reflectors for distributed denial-of-service attacks." In: *Computer Communication Review* 31.3 (2001).

[135] Hakem Beitollahi and Geert Deconinck. "Analyzing well-known countermeasures against distributed denial of service attacks." In: *Computer Communications* 35.11 (2012), pp. 1312–1332.

[136] Ayman Mukaddam, Imad Elhajj, Ayman I. Kayssi, and Ali Chehab. "IP Spoofing Detection Using Modified Hop Count." In: *Proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014)*. IEEE, 2014.

[137] Zakir Durumeric, Michael Bailey, and J. Alex Halderman. "An Internet-wide View of Internet-wide Scanning." In: *Proceedings of the 23rd USENIX Security Symposium (SEC 2014)*. USENIX Association, 2014.

[138] J. Postel. *Character Generator Protocol*. RFC 864 (INTERNET STANDARD). Internet Engineering Task Force, May 1983. URL: http://www.ietf.org/rfc/rfc864.txt.

[139] J. Durand, I. Pepelnjak, and G. Doering. *BGP Operations and Security*. RFC 7454 (Best Current Practice). Internet Engineering Task Force, Feb. 2015. URL: http://www.ietf.org/rfc/rfc7454.txt.

[140] Jaime Fink and Jack Manbeck. *Functional Requirements for Broadband Residential Gateway Devices*. Online: https://www.broadband-forum.org/technical/download/TR-124.pdf.

[141] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. "On the incompleteness of the AS-level graph: a novel methodology for BGP route collector placement." In: *Proceedings of the 12th ACM SIGCOMM Internet Measurement Conference, (IMC 2012)*. ACM, 2012.

[142] Ricardo V. Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. "The (In)Completeness of the Observed Internet AS-level Structure." In: *IEEE/ACM Transactions on Networking* 18.1 (2010), pp. 109–122.

[143] Ethan Katz-Bassett, Harsha V Madhyastha, Vijay Kumar Adhikari, Colin Scott, Justine Sherry, Peter Van Wesep, Thomas E Anderson, and Arvind Krishnamurthy. "Reverse traceroute." In: *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2010)*. USENIX Association, 2010.

[144] RIPE NCC. *RIPE Atlas: Internet data collection system*. Online: https://atlas.ripe.net/.

[145] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy H. Katz. "Towards an accurate AS-level traceroute tool." In: *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 2003)*. AMC, 2003.

[146] E. Rosen, A. Viswanathan, and R. Callon. *Multiprotocol Label Switching Architecture*. RFC 3031 (Proposed Standard). Updated by RFCs 6178, 6790. Internet Engineering Task Force, Jan. 2001. URL: http://www.ietf.org/rfc/rfc3031.txt.

[147] RIPE NCC. *RIPE Atlas: Statistics and Network coverage*. https://atlas.ripe.net/results/maps/network-coverage/.

[148] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. "Avoiding traceroute anomalies with Paris traceroute." In: *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference (IMC 2006)*. ACM, 2006.