EPFL

Thèse n° 10 247

Beyond the ratchet: practical challenges in secure messaging

Présentée le 10 janvier 2025

Faculté informatique et communications
Laboratoire de systèmes décentralisés et distribués
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Simone Maria Stefano COLOMBO

Acceptée sur proposition du jury

Prof. C. González Troncoso, présidente du jury
Prof. B. A. Ford, directeur de thèse
Prof. Ph. Jovanovic, rapporteur
Prof. S. Angel, rapporteur
Prof. A. Chiesa, rapporteur

École polytechnique fédérale de Lausanne

2025

A chi non ha mai smesso di lottare.
E a chi mi ha sempre sopportato.

Acknowledgements

I have been reflecting for quite some time on how to write this section of the thesis. It is difficult to properly acknowledge the immense help and support I have received—not only during the nearly six years of my PhD but also throughout my Bachelor's and Master's studies at EPFL. More than a decade has passed in the IN and BC buildings, and looking back with clarity is no easy task. Over these years, I have had the privilege of interacting with many people who helped me grow as a researcher, as well as those who supported and encouraged me outside of EPFL and academia. Without their presence and support, none of this would have been possible. In what follows, I will do my best to express my gratitude to the multitude of people who accompanied me on this journey—despite occasional distractions, like dealing with a landlord seemingly determined to test my patience at the very last moment.

First and foremost, I want to thank my advisor, Bryan Ford. Over these almost six years, Bryan granted me complete academic freedom to explore the topics that most interested me and to pursue both internal and external collaborations. These experiences taught me invaluable lessons and broadened my professional and personal horizons. From Bryan's research vision and intuition, I learned to see the bigger picture and to remain optimistic about my ideas and solutions, even when others, and even I, were not yet convinced. I am also deeply grateful for Bryan's understanding and support regarding the political restrictions I chose to place on funding and collaborations during my PhD. I will never forget Bryan's respect for my decision and his agreement not to pay me using funds I chose to avoid.

Next, I want to thank Henry Corrigan-Gibbs, whom I consider my unofficial co-advisor. Since when he spent one year as a postdoc in the DEDIS lab, Henry's support and guidance have been of fundamental importance to shape my view and approach on research. Henry also played an important role in extending my collaborations and discussions outside of EPFL, by introducing me to other researchers and colleagues during conferences, visits or during the mountain retreat after USENIX Security 2023. The long discussion we have had over the years have given me much and continue to do so.

I want also to express my gratitude to my private defense jury—Bryan, Carmela Troncoso, Alessandro Chiesa, Philipp Jovanovic and Sebastian Angel—for their valuable time, feedback and interesting discussion during the defense.

During my PhD, I had the privilege of collaborating with several coauthors, without whom I would never have been able to conduct my research as I did. In addition to Henry and Bryan, I want to extend my gratitude to Daniel Collins, Loïs Huguenin-Dumittan,

Khashayar Barooti, Serge Vaudenay, Kirill Nikitin, and David J. Wu. I learned so much from working with all of you.
EPFL has been my home for many years, and during this time, I have had the privilege of sharing it with numerous remarkable people. I would like to especially thank the former and current members, PhD students, engineers, and administrative staff of the DEDIS lab, as well as the past and present members of the LASEC lab, who welcomed me for collaborations and coffee breaks alike. I also want to express my gratitude to my classmates and teaching assistants from my Bachelor's and Master's studies at EPFL, with whom I shared both the joys and challenges of studying computer science and communication systems before embarking on my PhD journey.
It would not be true if I said that all these years were fun and games: some times were very challenging for various reasons. At EPFL and during our trips I had the chance to escape from the difficulties of research and PhD life in general with Andreas, Daniel, Loïs, Khashayar, Sylvain, Andi, Christian: thank you!
As I mentioned at the beginning of these acknowledgments, I have been fortunate to have many people by my side outside of research. Their presence beyond the confines of the academic ivory tower has been fundamental in helping me maintain my mental sanity and keep sight of the bigger reasons for why and how we do what we do. It would be impossible to name everyone here, and I value the importance of anonymity, but my gratitude extends to all my friends in Genève, Lausanne, Berlin, Bellinzona, Ticino, Catalunya, and beyond. To those who shared beers and laughter at the cinéma, BTR, and countless other bars, clubs, and homes in Genève and elsewhere. To the Lumpen Boys. To those who joined me in the boxing ring or went for a run when I needed to clear my mind. To those who carry TLA tattooed in their hearts and to those who have always struggled and continue to struggle. To all the people with whom I have shared streets and squares over the years. And to my family, whose unwavering support through the highs and lows of my PhD and life has been invaluable. Without all of you, this journey would have been entirely different—if not meaningless.
A questo punto è davvero tutto. Grazie a tutte le persone che mi hanno sopportato e supportato in questi anni.

Genève, 17 December 2024 Simone Colombo

Abstract

Secure messaging systems are essential for ensuring privacy and confidentiality in today's digital communication. Thanks to the widespread adoption of end-to-end encryption, messages are accessible only to intended users, and advancements in protocol resilience against secret compromise have enhanced messaging systems' protection guarantees. However, several open challenges remain. This thesis investigates three of these challenges—active attack detection, metadata protection during key retrieval, and real-world deniability—and presents cryptographic and system-level solutions to strengthen the security and privacy of modern secure messaging systems.

The first contribution of this thesis addresses active attack detection in messaging. We address scenarios where the network can delay and drop messages, and where adversaries can impersonate parties and inject forged messages. We propose out-of-band detection mechanisms that always detect active attacks, and in-band mechanisms that detect attacks as soon as an honest message goes through. Optimizing these schemes, we also explore how active attack detection can be practically achieved.

The second contribution addresses challenges in distributing cryptographic keys that enable parties to establish secure messaging channels. Metadata protection is crucial to safeguard users' social graphs, and security issues arise from potentially malicious service providers distributing adversarially-controlled keys. To address these challenges, we introduce authenticated private information retrieval, a cryptographic primitive that ensures clients 1) do not reveal their social graph to the messaging service and 2) either retrieve the correct key or abort. We implement and evaluate all our schemes, assessing the practicality of multi-server authenticated private information retrieval with Keyd, a PGP key-directory server we develop.

Finally, we analyze cryptographic deniability in secure messaging systems and its practical relevance from technical and legal perspectives. Although often presented as a key feature in protocols like Signal, our technical modeling, which incorporates real-world factors, along with legal analysis of 140 court cases in Switzerland, reveals that deniability typically fails in practice. Based on these findings, we discuss whether deniability is desirable and explore the challenges of designing systems that offer practical deniability.

Together, these contributions advance the resilience, privacy and practical applicability of secure messaging systems in the face of real-world adversaries.

Keywords: secure messaging, active attack, detection, private information retrieval, authenticated, deniability, real world, legal analysis, Signal

Sommario

I sistemi di messaggistica sicura sono cruciali per garantire la privacy e la riservatezza nella comunicazione digitale odierna. Grazie alla diffusione della crittografia end-to-end, i messaggi sono accessibili solo agli utenti coinvolti e la sicurezza di questi sistemi è assicurata dall'evoluzione della resilienza dei protocolli in caso di compromissione dei segreti crittografici. Tuttavia, rimangono ancora aperte numerose sfide. Questa tesi esplora tre di esse—il rilevamento degli attacchi attivi, la protezione dei metadati durante la distribuzione delle chiavi crittografiche e la negabilità (deniability) in contesti reali—proponendo soluzioni sia a livello crittografico che di sistema per migliorare la sicurezza e la privacy dei moderni sistemi di messaggistica.

Il primo contributo di questa tesi riguarda il rilevamento di attacchi attivi nella messaggistica. Ci focalizziamo su scenari in cui la rete può ritardare o bloccare messaggi e dove gli avversari possono impersonare gli utenti e inviare messaggi contraffatti. Proponiamo meccanismi di rilevamento fuori banda che identificano sempre gli attacchi attivi e meccanismi in banda che rilevano gli attacchi non appena un messaggio autentico viene ricevuto. Ottimizzando questi schemi, esploriamo anche come il rilevamento di attacchi attivi possa essere implementato concretamente.

Il secondo contributo affronta le sfide poste dalla distribuzione di chiavi crittografiche che consentono agli utilizzatori di stabilire canali di comunicazione sicuri. La protezione dei metadati è essenziale per proteggere i grafi sociali degli utenti, mentre la sicurezza è minacciata da fornitori di servizi di messaggistica potenzialmente malintenzionati che distribuiscono chiavi controllate da avversari. Per risolvere queste problematiche, introduciamo il recupero privato di informazioni autenticate (authenticated private information retrieval), un protocollo crittografico che garantisce che gli utenti 1) non rivelino i loro grafi sociali al servizio di messaggistica e 2) ricevano la chiave corretta o interrompano l'operazione. Implementiamo e valutiamo tutti i nostri schemi, testando la praticità del recupero privato di informazioni autenticate in un contesto multi-server con Keyd, un servizio di distribuzione per chiavi PGP da noi sviluppato.

Infine, analizziamo la negabilità crittografica nei sistemi di messaggistica sicura e la sua rilevanza pratica sia tecnica che legale. Nonostante sia spesso presentata come una caratteristica fondamentale di protocolli come Signal, il nostro modello tecnico, che integra fattori reali, e un'analisi legale di 140 casi giudiziari in Svizzera, indicano che la negabilità in generale fallisce. Basandoci su questi risultati, valutiamo se la negabilità sia auspicabile ed esploriamo le sfide nella progettazione di sistemi che possano renderla concretamente realizzabile.

Nel loro insieme, questi contributi migliorano la resilienza, la privacy e l'applicabilità concreta dei sistemi di messaggistica sicura a fronte di avversari reali.

Parole chiave: messaggistica sicura, attacco attivo, rilevamento, recupero privato di informazioni, autenticato, negabilità, casi reali, analisi legale, Signal

Contents

1 Introduction

> Problem selection is the most obvious aspect in determining our community's impact, and secure messaging, in all its forms, remains the most outstanding problem in crypto-for-privacy.
>
> Phillip Rogaway, *The Moral Character of Cryptographic Work* [Rog15, Part 4]

The ability to communicate securely and privately has always been crucial to human persona, social and political progress. From ancient methods of encoding messages to today's encrypted communication, the desire to protect privacy is universal—just as the attempts to breach it. With the rise of smartphones, communication has largely shifted to the digital realm, reshaping personal and professional interactions. Now, most of our communication travels through digital networks rather than on paper or via sound waves.

This shift has brought fast, convenient, and user-friendly communication methods, but it also presents significant security and privacy challenges. Fundamental properties of real-world in person communication—such as trust, privacy, ephemerality, deniability, protection from eavesdropping, metadata resistance, and ease of access—are hard to fully replicate in the digital world. These gaps create opportunities for surveillance and abuse by state and private actors alike.

Edward Snowden's revelations about the mass surveillance practices of the National Security Agency (NSA) and other governmental and private entities exposed widespread mass surveillance practices [Gua13, Gre14b]. This ignited a surge of interest and research into secure communication, particularly secure messaging. In an increasingly connected world, where governments and corporations have unprecedented access to personal data, the need for secure messaging solutions—and secure communication in general—has never been greater.

The key feature of modern secure messaging is *end-to-end encryption*: the sender encrypts messages that only intended recipients can decrypt. This ensure that if Alice starts a conversation with Bob, only Alice and Bob can read the messages. The messaging server operator, Internet service providers, telecommunication companies and malicious actors cannot access the content while the messages are in transit. End-to-end encryption, as opposed to client-to-server encryption, is now the default in most messaging solutions, largely thanks to the standard set by Signal.

The advent and widespread adoption of end-to-end encryption represent a major step forward for privacy and security, but significant challenges remain. Messaging conversations, which often last for extended periods, expose users to various types of attacks. A malicious actor may gain access to a device—during an arrest [ABJM21] or via malware like Pegasus [SRCM+22]—and clone the device's content to analyze past and future communications. Modern secure messaging solutions mitigate these risks by continuously updating key material, preventing adversaries from using cloned keys to decrypt prior or future messages. However, mechanisms for detecting such attacks, which would enable users to respond, are still lacking. Even if message content remains secure and active attacks like state exposure can be detected, metadata—information unrelated to the message's content—can still expose valuable insights to malicious actors. Details such as the time a message is sent or its recipient can reveal highly sensitive information. As Michael Hayden, former director of the NSA, stated during a debate at Johns Hopkins University, "We kill people based on metadata"[1]. Finally, even if the application encrypts conversations and protects metadata, malicious actors may still access plaintext messages, for example through coercion. Therefore, it is important to consider properties that protect message authors even after disclosure. One such property is deniability, which allows a party to plausibly deny authorship of a message. While deniability exists at the cryptographic level in several modern messaging solutions, its applicability to real-world scenarios remains questionable.

This thesis contribute to these three practical challenges in secure messaging. First, we propose cryptographic solutions to detect active attacks on secure messaging solutions. Next, we focus on metadata privacy by introducing a new cryptographic primitive that enables users to privately and securely retrieve public encryption keys of their communication partners. Finally, we examine the limitations of cryptographic deniability and propose ways to bridge the gap between theory and real-world applicability.

In the next section, we provide an overview of the thesis. This is followed by the necessary background and a technical summary of our results. We conclude the introduction with bibliographic notes and a list of the notation used throughout the thesis.

[1]The full debate is available on YouTube and the sentence that we cite is pronounced at 17:59 (https://www.youtube.com/watch?v=kV2HDM86XgI).

1.1 Thesis overview

In this section we present a high level overview of the thesis.

In Chapter 2, we focus on cryptographic protocols, particularly on active attack detection in messaging with immediate decryption. We consider an adversary that can impersonate parties and inject messages on their behalf. If the adversary continues impersonating a party indefinitely, in-band detection is impossible. However, if the compromised party regains control, e.g., due to non-persistent malware like some versions of NSO's Pegasus, detection mechanisms can be deployed. Immediate decryption [ACD18, ACD19], which allows messages to be dropped or reordered at the protocol level without stalling future communication, adds complexity to both analysis and design. Out-of-order messages must not be mistaken for adversarially injected messages and detection mechanisms must handle dropped messages. We propose solutions for both in-band and out-of-band detection and explore practical optimizations and performance/security trade-offs.

In Chapter 3, we explore the integration of cryptographic protocols into messaging systems. We propose novel approaches to defining maliciously-secure private information retrieval and applies them to public-key servers. Private information retrieval (PIR) [CGKS95] enables a client to fetch a record from a database while hiding the query from the database server(s), but most existing schemes do not protect data integrity in the presence of a malicious server. If a client uses PIR to query a public-key server, a malicious server could force the client to fetch a false public key for which the adversary controls the secret key. We address this issue by defining authenticated PIR: the client either retrieves the correct database record or aborts. We present single- and multi-server schemes and evaluate their performance with PGP key-directory servers.

In Chapter 4, we analyze these systems and how their properties integrate in the real world. To this end, we analyze whether cryptographic deniability holds in the real world, with a particular focus on messaging. We adopt an approach that combines cryptographic and legal analysis. We propose a formal model that considers the entire communication system to analyze deniability in practice and apply it to the Signal application and DKIM-protected email. Our findings demonstrate that these systems do not offer practical deniability guarantees. We also analyze 140 court cases in Switzerland where conversations from messaging applications were used as evidence. None of these cases consider deniability, suggesting that this property does not impact the legal setting. Based on these findings, we assess whether deniability is a desirable property and the challenges and shortcomings of designing a system that is deniable in practice.

We continue this introduction with a general background on secure messaging and then introduce the three chapters that compose this manuscript and their respective contributions.

1.2 Background

In this section, we provide an overview of secure messaging, focusing on three concentric layers: cryptographic protocols, systems and their interaction with the real world.

Protocols. At the core of modern secure messaging solutions are cryptographic protocols, which enable parties to exchange messages while ensuring various cryptographic guarantees. These guarantees include traditional properties of secure communication systems, providing confidentiality, integrity, and authenticity: only the intended recipients can read the messages, which cannot be altered and must come from the correct sender. Beyond these classic properties, modern secure messaging protocols address more nuanced concerns.

Given the prevalence of surveillance [BSJ+15] and the fact that messaging sessions usually spans years researchers and practitioners have had to address state *exposure* or *compromise* [CCG16] when designing cryptographic protocols. Two key security concepts that mitigate state exposure are *forward secrecy* and *post-compromise security*. Forward secrecy [Gün89, BG21] ensures that even if a party's secret state is compromised, previous encrypted conversations remain secure. Post-compromise security [CCG16] ensures that, even after an attacker compromises a messaging session, the security of future communications can be restored. The cryptographic protocols achieve these properties by regularly updating, or *ratcheting*, the cryptographic keying material [BGB04, BSJ+17].

Some of these protocols also ensure additional properties [UDB+15]. For instance, *deniability* [DNS98] enables parties to plausibly deny their participation in a protocol execution, or authorship of a message. *Immediate decryption* [ACD19], which ensures that the protocol handles dropped or out-of-order messages, enhances the protocol's resilience to network failures and enable bandwidth optimizations, for example by buffering large messages. This thesis explores some of these properties and their interaction with messaging systems and real-world conditions.

The de-facto standard [EM19] protocols that ensure these properties are Signal's key exchange protocol X3DH [MP16] and the Double Ratchet algorithm [PM16], both of which are used in applications like Signal, WhatsApp, Wire, Skype and Facebook Messenger's "Secret Conversations" feature. X3DH, the extended triple Diffie-Hellman protocol that builds on the triple Diffie-Hellman [BJM97], is an authenticated key exchange protocol [DvOW92] that enables two parties to securely establish initial keying material for the messaging session while authenticating the parties. This protocol uses a public-key infrastructure to establish a channel even when one party is offline. The Double Ratchet, descendant of the Off-The-Record (OTR) protocol [BGB04], enables parties to encrypt messages while updating their keys. Forward secrecy builds on a symmetric hash-based ratchet, while the protocol ensures post-compromise security using an asymmetric ratchet, which operates as a continuous Diffie-Hellman key exchange [DH76].

Systems. The cryptographic protocols that ensure the security properties discussed in the previous paragraphs do not operate in isolation but are part of a larger messaging *system*. The transition from cryptographic protocol to full messaging systems introduces new security and privacy risks, as well as usability challenges. A typical messaging session between two parties—which is the focus of this thesis—involves multiple actors: the two participants (Alice and Bob), their devices and the (logical) server that manages the public key infrastructure and routes the messages. The interaction between these actors introduce new security and privacy risks that cryptographic protocols alone are not designed to address, as they typically focus on securing communication between parties but may not account for system-level vulnerabilities like server trust and device management [MP17, CFKN20, CDDF20, CJN23].

One notable example is the security risks posed by the PKI. When Alice wants to initiate a conversation with Bob, she queries the server for Bob's public key. A malicious server could respond with a false public key for which the adversary controls the secret key, thereby enabling the adversary to impersonate Bob. Different approaches have been proposed to mitigate this risk. Signal's safety numbers protocol [Mar17b] enables parties to compare the long-term keys provided by the PKI through an out-of-band channel, such as scanning a QR code. Assuming that the out-of-band channel provides authenticity, this enables parties to audit the PKI and prevent these impersonation attacks. A fruitful line of work [MPC+18, TGL+19, CDGM19, CDG+22, MKS+23] initiated by the authors of CONIKS [MBB+15], proposes ensuring consistency for the PKI's bindings using ideas adapted from transparency log systems [Lau14, Rya14]. However, these approaches also rely on out-of-band channels, with the associated usability challenges [KFR09], and do not address a critical issue related to the PKI's role in messaging: privacy.

Privacy is also compromised during the key retrieval process. When Alice fetches Bob's key, she informs the server of her intention to communicate with Bob, enabling the server to construct Alice's *social graph*—a detailed view of her interactions via the messaging application [NS09, BDK11, NSR11]. Solutions to address this and other privacy challenges that arise during data queries usually relies on cryptographic primitives such as PIR or hardware enclave technology [MPC+18, DFD+21]. However, these solutions are in some cases costly to deploy and fail to address the key binding issue discussed in the previous paragraph.

The PKI privacy issue mentioned above is just one example of *metadata leakage*, a broader challenge that many secure messaging systems continue to face. Metadata, unlike the content of messages, reveals information about communication patterns, such as who is communicating with whom and when. Metadata are leaked during various processes, such as contact discovery [KRS+19, HWS+21, HSW23, MSGJ24] or authentication with the messaging server [Lun18]. As we will explore in this thesis, the leakage of such metadata can undermine the security and privacy properties provided by cryptographic protocols.

Real world. Every secure messaging system interfaces with the real world, and the security properties it provides—or claims to provide—must hold under real-world conditions. This interaction is multifaceted, encompassing security, privacy, usability, legal considerations, user interaction challenges and other aspects. Each of these dimensions presents unique challenges that must be addressed to ensure the system's overall robustness. It is equally important to consider both the technology and the social dynamics it serves, the power relationship it creates, and the contexts from which it stems. Secure messaging systems, as every computer system, do not exist in a vacuum: they are influenced by the social, economic, and political environments in which they are create and in which they operate. Neglecting these factors may lead to incomplete and ineffective solutions at best or draconian ones at worst [GGV20, AJ20, AAB+24, Vau22].

Studying secure messaging systems, and compute systems more broadly, in relation to society often requires multidisciplinary approaches, which are increasingly prominent in our fields (cf. [FPS+18, AJ20, SV21, ABJM21, DSKB21, RMA+23, YGS23] for a non-exhaustive list). One example of such an approach is the combination of cryptographic and legal analysis to evaluate how a cryptographic property (does not) apply in the real world.

First, the security and privacy guarantees offered by cryptographic properties, which are designed to function within specific boundaries (i.e., under a defined threat model), may break down when exposed to more sophisticated and powerful real-world adversaries. For example, classic information retrieval protocols do not provide integrity guarantees if the server is malicious, or, from a system perspective, a malicious server managing a public-key infrastructure can mount impersonation or MITM attacks by serving wrong public keys. This gap between the theoretical security properties and the practical reality must be acknowledged and addressed in the design and deployment of secure messaging systems.

Second, information security interacts with and is influenced by various external factors, particularly legal and regulatory frameworks. The legal environment plays a crucial role in shaping how cryptographic systems are perceived and used [Bla12], and, similarly, advances in information security can have a positive impact on policies and state-controlled mass surveillance. Surveillance and abuse by state and private actors often occur within or are enabled by legal mechanisms, making it essential for security properties to be recognized not only in technical terms but also in legal contexts. For instance, encryption and cryptographic protections may be undermined by legal mandates, such as subpoenas, key disclosure laws or backdoor requirements. Therefore, security properties must be robust enough to resist not only technical attacks but also legal exploitation.

1.3 Technical overview

In this section we outline the three main contributions of this thesis.

Active attack detection in messaging with immediate decryption

Forward secrecy and post compromise security protect cryptographic protocols against state exposure, but do not address the *detection* of such attacks. If an adversary compromises the PKI, it can impersonate one of the parties. To mitigate this several messaging systems enable parties to verify their long-term keys using an out-of-band channel. However, if the adversary compromises the session after key establishment—during the Double Ratchet protocol—existing out-of-band detection mechanisms are insufficient. In such cases, the adversary can mount an active attack and impersonate parties, even if the PKI remains uncompromised. Detecting these active attacks requires an authentic out-of-band channel. Without it, the adversary can block all honest messages that a compromised party sends and replace them with forged ones. Solutions have been proposed to address this, but they either fail to fully authenticate the conversation transcript [DH21] or require multiple rounds of interaction without formally supporting immediate decryption [DGP22].

Out-of-band channels are not always available and even when they are, they may reduce usability or be underutilized [KFR09]. Durak and Vaudenay [DV19] propose a compromise: if a single honest message can pass through after a compromise, parties can detect active attacks using the same in-band channel that they already use for message exchange. In other words, once an honest message reaches its destination, in-band active attack detection becomes possible. They introduce RECOVER security, later extended by Caforio et al. [CDV21], which enables active attack detection under these conditions. RECOVER security builds on two complementary notions: r-RECOVER ensures that a party receiving a forged message will reject all subsequent honest messages from the counterpart, effectively aborting communication as soon as the first honest message is received. The complementary notion, s-RECOVER, guarantees that a party must not accept messages from someone that has received a forgery. These notions and corresponding constructions, however, do not support immediate decryption, making them unsuitable for systems like the Signal application that rely on the Double Ratchet protocol.

Chapter 2 addresses these issues by exploring active attack detection for messaging with immediate decryption, considering both in in-band and out-of-band settings. We extend the work of Durak and Vaudenay, and Caforio et al., introducing two security notions (r-RID and s-RID) that generalize the r-RECOVER and s-RECOVER to support immediate decryption. We then present a construction that builds on a classic messaging scheme and is secure under both notions. Additionally, we introduce two analogous security notions (r-UNF and s-UNF) for the out-of-band setting, where parties use an authentic

out-of-band channel, and we design schemes that are secure under these notions. The extended abstract [BCC+23a] on which Chapter 2 demonstrated that satisfying r-RID and r-UNF security notions causes the ciphertext size to grow linearly in the number of messages sent and security parameter in the worst case. This result implies that our naïve construction, which attaches all previously sent ciphertexts to each new message to achieve r-RID security, is asymptotically optimal. The same applies to r-UNF in the out-of-band setting. Given this inefficiency, we show that the complementary notions, s-RID and s-UNF, impose a lower overhead and can be sufficient in practice, as they provide r-RID-like and r-UNF-like guarantees after an honest round-trip. We propose several optimizations, culminating in a scheme whose communication overhead for s-RID security is reduced to a single hash digest and a few message indices, with the exact number depending on how synchronized the communication is. Since s-RID security provides r-RID-like guarantees after an honest round trip, we posit that this solution is efficient enough for practical adoption. In the out-of-band setting we propose a three-move authentication protocol that similarly reduces communication overhead.

Authenticated private information retrieval

The cryptographic protocols that ensure the security properties discussed in the previous section do not operate in isolation but are part of a larger *system*.

In Chapter 3, we address these security and privacy issues by introducing *authenticated private information retrieval* and applying the schemes of this new cryptographic primitive to a PGP key directory server. While this chapter shifts focus from messaging to PGP, which is primarily used to secure email communications, the principles of authenticated PIR can be applied in various settings.

Private information retrieval (PIR) [CGKS95] enables a client to fetch a record from a database while hiding from the database server(s) which specific record(s) the client retrieves. Beyond its use in privacy-preserving key retrieval, PIR has numerous other applications, such as metadata-private messaging [AS16, ACLS18], certificate transparency [LG15, Rya14], video streaming [GCM+16], password-breach alerting [TPY+19, LPA+19, ALP+21], retrieval of security updates [Cap13] and private SQL-like queries on public data [OG10, WYG+17]. However, most PIR protocols do not ensure data authenticity in the presence of malicious servers. In many multi-server PIR schemes [CGKS95, BGI16], a single adversarial server can corrupt the client's output by flipping any subset of bits. In all single-server PIR schemes [KO97] we know of prior to the publication of the extended abstract [CNCG+23a] on which Chapter 3 builds (c.f., [KO97, CMS99, Lip05, AMBFK16, PPY18, BIPW17, ACLS18, GH19, CK20, PT20, ALP+21, MCR21, MW22, HHCG+23, DPC23] for a non-exhaustive list), a malicious server can choose the exact output that the client will receive by substituting all the database records with a chosen record before processing the client's request. In applications where data integrity

matters, such as a PGP public-key directory, unauthenticated PIR is inadequate. While PIR can help Alice retrieve Bob's public key without disclosing this information to the PKI, it cannot prevent a malicious PKI from returning a false public key.

Authenticated private information retrieval augments the standard privacy properties of classic PIR with strong—in a cryptographic sense that we precisely define—authenticity guarantees. In the multi-server setting, we propose authenticated-PIR schemes for:

- *Point queries*, in which a client wants to fetch a particular database record. For example, "What is the public key for `user@epfl.ch`?"

- *Predicate queries*, where a client wants to apply an aggregation operator—such as `COUNT`, `SUM`, or `AVG`—to all records matching a predicate. For example, "How many keys are registered for email addresses ending in `@epfl.ch`?"

Our multi-server authenticated-PIR schemes guarantee integrity as long as at least one of the PIR servers in honest, i.e., in the *anytrust* model [WCGFJ12]. In contrast, prior work addressing malicious of faulty PIR servers in the multi-server setting either requires a majority or supermajority of honest servers [BS02, BS07, Gol07, DGH12] or relies on expensive public-key cryptography operations [ZS14]. Our multi-server scheme, however, use only fast symmetric cryptography and ensure authenticity through two different mechanisms. The scheme for point queries carefully combines a Merkle tree and a classic PIR scheme to enable the client to verify the servers' responses. The scheme for predicates queries uses function secret sharing [BGI15, BGI16, WYG$^+$17] and information-theoretic message authentication codes [CDF$^+$08, DPSZ12] to provide integrity.

In the single-server setting, we introduce authenticated-PIR schemes for point queries which provide authentication as long as the client can obtain a short digest of the database via out-of-band means. We propose two different schemes that rely on the decisional Diffie-Hellman [Bon98] and learning with errors assumptions [Reg05]. Both schemes extend the classic Kushilevitz-Ostrovsky scheme based on additively homomorphic encryption [KO97, OS07], and operate on single-bit records, though we propose extensions for handling larger records at the price of increased client computation.

In both settings, privacy and authenticity holds also in the presence of selective-failure attacks by malicious servers [HKE13, KS06, KO97]. In such attacks, a malicious server answers the client's query with respect to a database that differs from the true database in a few rows. By observing whether the client accepts or rejects the resulting answer, the server can learn information about which rows the client had queried. To defend against these attacks, our security definitions require that any misbehavior on the part of a malicious server causes a client to reject the servers' response

Real-world deniability in messaging

In Chapter 4, we analyze whether cryptographic deniability in messaging holds in the real world. Deniability, according to the online dictionary `www.dictionary.com`, is "the ability to deny something, as knowledge of or connection with an illegal activity". Despite the negative connotation, this definition captures the coarse notion agreed upon by researchers and practitioners: deniability enables a user to *plausibly deny* their involvement in executing some scheme or protocol. The Signal protocol [MP16, PM16]—which encompasses X3DH and the Double Ratchet—claims to offer deniability. Moxie Marlinspike, one of the designers of Signal, discusses deniability in the context of OTR [BGB04] as follows [Mar13]:

> "If someone receives an OTR message from you, they can be absolutely sure you sent it (rather than having been forged by some third party), but can't prove to anyone else that it was a message you wrote."

In the cryptographic literature, deniability is typically formalised as a game played between abstract entities. A protocol is deniable if a judge cannot differentiate between a real execution of the protocol and a simulated one. However, considering what we discussed before, it is crucial to question whether cryptographic deniability (1) holds *technically* in a complex ecosystem such as Signal, and (2) has a tangible impact in the real world, such as in a court of law [Gre14a]. Chapter 4 analyzes these two parallel aspects of deniability.

On the technical side, we propose a new model for deniability in secure messaging, that captures the fact that, in practice, messages are routed between users via a server that usually authenticated users. In our model, the judge receives Bob's state (e.g., their entire phone or screenshots of the conversation) after allegedly communicating with Alice. The judge also has data from the server and any other relevant information available. We posit that the system is deniable if there exists a practical simulator who, under application-specific constraints, can interact with the server and simulate a state that is indistinguishable from Bob's. This approach extends the classical notions, which only consider the cryptographic transcript, by providing the judge with Bob's state, which can include his entire phone, a portion of the server's state and arbitrary auxiliary data [RGK06]. Our model broadens the purely cryptographic approach incorporating real-world evidence and higher-level components that can undermine deniability. We apply this model to two real-world applications: Signal, and KeyForge [SPG21], a solution for deniability of email with DomainKeys Identified Mail (DKIM) protection [CHK11]. Using our model, we show that Signal is not deniable in practice and that KeyForge also poses challenges to practical deniability.

On the legal side, we analyze 140 legal cases in Switzerland that uses WhatsApp conversations as evidence. Since WhatsApp uses the same core protocol as Signal for two-party

messaging, these conversations are cryptographically deniable. We find that (1) in only two cases the legitimacy of such evidence is questioned, (2) judges always accept this evidence, even if disputed, and (3) no case mentions or considers deniability. Although our findings cannot be generalized to other countries, our analysis suggest that cryptographic deniability does not hold up in a legal setting.

Both technical and legal analysis show that cryptographic deniability is ineffective in the real world. In the last part of Chapter 4 we discuss whether deniability should be a goal of messaging solutions by analyzing the issues and shortcomings that practical deniability brings. Given our model to analyze deniability and the analysis of technical and legal limitations, we claim that deniability should either not be a goal of messaging solutions or these must aim for practical real-world deniability. We argue that for deniability to be practical, it must be easily accessible to all users. Under our notion, Signal would achieve deniability if the application allowed users to modify, insert or delete messages stored on their devices, enabling all users to simulate conversations in practice. If deniability is a goal, we advocate to implement message modification on the local device, in Signal and other secure messaging applications. We also discuss the risks that such editing capabilities entail.

1.4 Bibliographic notes

This thesis is based on parts of the following jointly authored publications. Where appropriate we cite also the full versions.

Chapter 2: K. Barooti, D. Collins, S. Colombo, L. Huguenin-Dumittan, S. Vaudenay. On Active Attack Detection in Messaging with Immediate Decryption. *CRYPTO*, 2023 [BCC$^+$23a, BCC$^+$23b];

Chapter 3: S. Colombo, K. Nikitin, H. Corrigan-Gibbs, D. J. Wu, B. Ford. Authenticated private information retrieval. *USENIX Security*, 2023 [CNCG$^+$23a, CNCG$^+$23b];

Chapter 4: D. Collins, S. Colombo, L. Huguenin-Dumittan. Real-World Deniability in Messaging. *PETS*, 2025 (to appear) [CCHD25, CCHD23].

1.5 Notation

In this section we present the notation that we will use throughout this thesis.

Sets. We use $\mathbb{N}$ to denote the set of natural numbers. For $N \in \mathbb{N}$, $[N] = \{1, \ldots, N\}$. For a finite set S, we write $x \leftarrow\!\!\$\ S$ to indicate that x is sampled independently and uniformly at random from S. Given a set S, S^* (respectively S^n) is the set of all strings of arbitrary length (resp. of length n) whose elements are in S. We denote the empty string by ε. For finite sets S and T, we use $\mathsf{Funs}[S, T]$ to denote the set of all functions from S to T.

Algebra. We denote with $\mathbb{Z}_m$ the ring of integers modulo m. We use $\mathbb{F}$ to denote a finite field. We will typically take $\mathbb{F}$ to be the set of integers modulo a prime p with addition and multiplication modulo p. For a group $\mathbb{G}$, we use $1_{\mathbb{G}}$ to denote the identity element.

Vectors. For vectors $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$ and $y = (y_1, \ldots, y_n) \in \mathbb{F}^n$, we use $\langle x, y \rangle$ to denote their inner product $\sum_{i=1}^{n} x_i y_i \in \mathbb{F}$. We denote with $e_i \in \mathbb{F}^n$ the unit vector of length n, which is zero everywhere except at a single coordinate i, where it has value 1.

Indistinguishability and algorithms. We use $:=$ to define a new function or symbol. We use $\mathsf{negl}(\cdot)$ to denote a negligible function and $\mathsf{poly}(\cdot)$ to denote a fixed polynomial. The symbol $\perp$ is an output that indicates rejections. PPT abbreviates "probabilistic polynomial time", which we use in the context of algorithms bounded in terms of the security parameter λ. By "efficient algorithm" we refer to a probabilistic polynomial time algorithm.

Maps. We use maps, or associative arrays, to associate keys with values. A map is initialized as $m[\cdot] \leftarrow x$, where all values are initially set to x. The expression $m[k]$ returns the element indexed by key k. Keys can be tuples of any length $n \geq 1$. We index maps with integers starting from 1; in this case, $m[a : b]$ returns the list of elements whose keys fall between a and b. Tuple elements are accessed using dot notation. The function $\mathsf{length}(m)$ returns the total number of keys in the map m.

Probability distributions. We use $\mathsf{SD}(\cdot, \cdot)$ to denote the statistical distance between two distributions. We write $\mathcal{D}_0 \approx_c \mathcal{D}_1$ to denote that the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable.

Participants. In Chapters 2 and 4 we consider two parties that take part in a protocol. We consider in this case Alice and Bob, which we denote as A and B. Let $\mathcal{P}$ be one party (A resp. B) and $\overline{\mathcal{P}}$ be their partner (B resp. A).

2 On active attack detection in messaging with immediate decryption

The widely used Signal protocol provides protection against state exposure attacks through forward and post-compromise security but does not enable parties to detect such attacks. Additionally, it supports *immediate decryption*, allowing messages to be re-ordered or dropped at the protocol level without impacting correctness. In this chapter, we consider *active attack detection for secure messaging with immediate decryption*, enabling parties to detect active attacks instantly under certain conditions. We first consider in-band active attack detection, where compromised participants can detect an attack if they are able to send a single message to their partner. We propose two complementary notions to capture security, and present a compiler that provides security with respect to both notions. Our notions generalise existing works (RECOVER security [DV19, CDV21]) which only supported in-order messaging. We also explore out-of-band attack detection by leveraging out-of-band authenticated channels and propose analogous security notions. The extended abstract on which this chapter builds [BCC+23a] shows that one of our two notions in each setting imposes a linear communication overhead in the number of sent messages and security parameter. This implies that each message must information-theoretically encapsulate all previous messages, making our naïve construction, which effectively attaches the entire message history to each new message, asymptotically optimal. We then explore ways to bypass this lower bound and highlight the feasibility of practical active attack detection compatible with immediate decryption.

An extended abstract corresponding to this work appeared at CRYPTO 2023 [BCC+23a], with a full version available on the Cryptology ePrint Archive [BCC+23b]. The work presented in this chapter results from a close collaboration with Daniel Collins, with additional contributions from Khashayar Barooti, Loïs Huguenin-Dumittan and Serge Vaudenay. The author of this thesis significantly contributed to the definition of (authenticated) ratcheted communication and the corresponding security properties, as well as to the design of schemes for out-of-band active attack detection and their optimizations. Due to the close collaboration, much of the content in this chapter also appears in Daniel Collins' PhD thesis [Col24]. Daniel Collins is the primary contributor to the epoch-based optimization presented in Section 2.5.3, which is included here for completeness.

2.1 Introduction

As highlighted in Chapter 1, since the Snowden revelations and given the unprecedented rise of mass surveillance, many messaging solutions have strengthened their security guarantees. The vulnerability to state exposure attacks has motivated researchers and practitioners to develop ratcheting-based schemes, which ensures forward secrecy—protecting the confidentiality of messages sent prior to state exposures—and post-compromise security—enabling parties to automatically restore confidentiality after a compromise [CCG16].

The asynchronicity of messaging and the unreliability of certain network protocols have further driven the design of ratcheting-based schemes with *immediate decryption* [ACD19, PP22, CZ22]. These schemes support out-of-order delivery and handle message loss at the *protocol level*, ensuring that receivers can decrypt legitimate messages as soon as they arrive and place them correctly among other received messages, even if earlier messages are delayed. Furthermore, communication can continue even if some messages are permanently lost—a feature Alwen et al. term message-loss resilience [ACD19, Section 1]. As discussed in Chapter 1, the Signal protocol [PM16] supports immediate decryption, whereas many other schemes in the literature fail when even a single message is lost. (see [BSJ+17, PR18a, DV19, CDV21, CGCD+17] for a non-exhaustive list).

The aforementioned security notions do not guarantee message authentication when the adversary impersonates parties, e.g., through state compromise. The lack of authentication implies that parties cannot *detect* active attacks. A recent phishing attack against Signal's phone number verification service enabled attackers to re-register accounts to another device, demonstrating the practicality of impersonation attacks via secret state compromise [Sup22]. Similar attacks that steal verification codes to hijack accounts affect a plethora of messaging applications. The proliferation of spyware such as NSO Group's Pegasus represents an additional—and worrying—threat for secret exfiltration [SRCM+22].

The most widely used mechanisms for detecting active attacks rely on an *out-of-band* authenticated channel. All such mechanisms we are aware of, whether deployed in practice [Mar17b] or proposed in the literature [DH21, DGP22] assume the availability of such a channel. Solutions like Signal's safety numbers [Mar17b] enables parties to authenticate long-term keys distributed by the Signal server by comparing a sequence of numbers or QR codes in person. However, Dowling and Hale [DH20, DH21] point out that Signal's approach—and all similar methods to our knowledge—does not provide any guarantees after a user's state is compromised, since safety numbers only authenticate initial keys (in Signal's case, the keys that the X3DH key agreement protocol generates). Dowling and Hale remark on Signal's approach [DH20]:

> "A successful verification indicates that an attacker has not modified the long-term keys of either party in the session. However, as the verification is tied only to long-term keys and not the current session state, there is no indication of who the current communication partner really is. Following a compromise, an attacker can send/receive messages and impersonate a partner device, and this will not be detected during the Signal entity authentication protocol."

To address this issue, Dowling and Hale [DH21] proposed adding an additional authentication key to each iteration of Signal's asymmetric ratchet for on-demand use in out-of-band authentication. Their construction enables parties to immediately—that is, without additional communication rounds—authenticate their entire *asymmetric ratchet* out of band. However messages forged under symmetric keys will never be detected. The only other construction in the literature to our knowledge, proposed by Dowling, Günter and Poirrier [DGP22], requires *three rounds* of in-band communication before an out-of-band hash comparison can take place. Unlike Dowling and Hale's solution, this approach authenticates all messages (though does not formally treat out-of-order messages), but the additional communication rounds pose challenges, especially in the presence of an active adversary. This brings us to our first research question:

1. Can we authenticate *all* messages in a *single round* of out-of-band communication to detect active attacks in the immediate decryption setting?

Out-of-band authentication is not always practical or even possible [KFR09]. A convenient alternative is to detect active attacks *in-band*, i.e., using the same channel as the messaging protocol. The adversary can, in the worst case, block all messages sent by honest parties, thereby forcing users to resort to out-of-band communication, but mounting such a persistent attack requires considerable resources. Durak and Vaudenay [DV19] introduce RECOVER security to model in-band active attack detection: if a party receives a forgery, then this party does not accept subsequent messages sent honestly by his counterpart. Caforio et al. [CDV21] extend RECOVER security to enable a party to detect whether their *partner* was compromised, i.e., whether they received a forgery. By contrast to out-of-band authentication, no additional messages are required to support attack detection: in-band ciphertexts contain the authentication information. However, these notions and the corresponding constructions assume in-order message delivery and fail on message dropping. This raises a second question, first suggested by Alwen et al. [ACD18, ACD19]:

2. Can we achieve extended RECOVER security—immediate in-band active attack detection—while supporting immediate decryption?

To detect active attacks, parties need to authenticate their *entire* message history: each message may be a forgery, i.e., the result of an active attack. With immediate decryption,

parties cannot be sure which messages their partner has received until they receive an honest reply from them. Intuitively, each message needs to "contain" the message history up until when it was sent. The extended abstract corresponding to the content of this chapter proves this intuition [BCC+23a, Section 6], which motivates the exploration of performance/security trade-offs and optimisations. In this regard, existing protocols for both in-band and out-of-band active attack detection represent only a subset of the potential design space. Consequently we also ask:

3. What are the *communication costs* of in- and out-of-band active attack detection for messaging with immediate decryption, and what useful performance/security *trade-offs* can be made?

Technical overview

We assume a network where parties communicate over two types of channels: insecure channels and out-of-band authenticated channels. The adversary has full control over insecure channels, allowing her to read, deliver, modify and delay messages. In the authenticated channels, however, the integrity and authenticity of the messages are protected, meaning the adversary can still read, deliver, duplicate, and delay messages but cannot modify them. In the Signal application, the insecure channel corresponds to the usual network, while the out-of-band channel is used for safety number verification [Mar17b], typically conducted in person.

(Authenticated) ratcheted communication. We introduce a syntax for ratcheted communication (RC) in which sent and received messages are associated with totally ordered *ordinals* (epoch/index pairs in Signal [ACD19]). Ordinals enable our protocol to support *immediate decryption* [ACD19], i.e., message loss and re-ordering on the network. We build on this syntax to define *authenticated ratcheted communication*, or ARC, which comprises two additional functions AuthSend and AuthReceive. A party can use AuthSend to send an authentication tag through the out-of-band channel that the counterpart processes with AuthReceive. AuthSend outputs an ordinal that is at least as large as the last *sent* ordinal for that party. AuthReceive, if successful, should authenticate all messages up to that ordinal; this is captured in UNF security. Our notion ORDINALS enforces these semantics even in presence of adversarial forgeries.

RID security. We revisit the definitions of RECOVER security [DV19, CDV21] in the immediate decryption setting. We define two complementary security notions for RID security:

- r-RID ensures that the receiver of a forgery does not accept honest messages with ordinals *larger* than that of the forgery.

- s-RID security enables a party to detect if their counterpart has ever received a forgery (i.e., the sender was victim of a forgery before sending the honest message).

Immediate decryption imposes particular care in defining the two notions: parties cannot rely on messages' order of arrival to detect forgeries. If a scheme is both r-RID-secure and s-RID-secure, then it is RID-secure (recover with immediate decryption). These notions are orthogonal to forward security and post-compromise security guarantees in secure messaging.

We propose a construction that transforms any ratcheted communication scheme into a provably RID-secure one. In the construction, both parties keep track of messages they have sent and received. Every time they send a message, they attach *all* messages (i.e., the ciphertexts from the underlying RC) they have sent and received so far to their ciphertext. When a party receives a message that "contradicts" what it has sent or received, it can deduce that an active attack took place.

To reduce bandwidth, parties send ordinals and hashes of messages, instead of complete ciphertexts. For r-RID security, a receiver $\overline{\mathcal{P}}$ compares the input message and the sender $\mathcal{P}$'s supposed set of sent messages with what $\overline{\mathcal{P}}$ has received previously. For s-RID, it suffices for a receiver $\overline{\mathcal{P}}$ (who knows exactly what it sent) to check whether the sender claims to have received anything that $\overline{\mathcal{P}}$ did not send. Here, $\mathcal{P}$ only needs to send a *single* hash alongside the set of received ordinals (which are generally smaller than hashes), since $\overline{\mathcal{P}}$ can recompute the hash locally. Since the channel is insecure, parties need to perform a series of checks on the ciphertexts to prevent the adversary from tampering with the sets of sent and received messages sent. Both r-UNF- and s-UNF-security build on the collision resistance of the hash function.

UNF security. We define notions analogous to r-RID and s-RID for *authenticated* ratcheted communication schemes. i.e., for schemes that use both in-band and out-of-band authenticated channels. The r-UNF (receiver unforgeable) notion ensures that a party does not accept authentication tags after receiving a forgery, whereas s-UNF (sender unforgeable) ensures that a party does not accept authentication tags coming from a counterpart that received a forgery. If a scheme is both r-UNF-secure and s-UNF-secure, then it is UNF-secure. We show that a RID-secure scheme can be turned into a UNF-secure scheme. The transformation highlights the similarity between RID and UNF security. In the former, parties authenticate all messages they have sent and received in band, whereas in the latter the messages are authenticated out of band. Concretely, the transformation uses the ciphertext of a RID-secure RC scheme as the authentication tag for an ARC scheme, by moving authentication material to the out-of-band channel.

Practical active attack detection. We explore how to overcome the linear communication complexity that r-RID and r-UNF impose [BCC+23a]. Notably, we observe that s-RID

and s-UNF can provide r-RID/r-UNF-like guarantees after a single communication round. If $\mathcal{P}$ detects that their partner $\overline{\mathcal{P}}$ has received a forgery, $\mathcal{P}$ can notify $\overline{\mathcal{P}}$, allowing $\overline{\mathcal{P}}$ to learn that they have received a forgery, effectively achieving r-RID/r-UNF guarantees.

We demonstrates that protocols can achieve s-RID/s-UNF security at a significantly lower cost than r-RID/r-UNF. A party $\mathcal{P}$ needs only to send a single hash of received messages along with corresponding ordinals, since partner $\overline{\mathcal{P}}$ can recompute the hash if it stores all previously sent messages.

To further reduce communication overhead, we introduce the use of *epochs* [ACD19], similar to Signal's Double Ratchet [PM16]. Each party starts with with $\mathsf{epoch}_{\mathcal{P}} = 0$ $\mathsf{epoch}_{\overline{\mathcal{P}}} = 1$, respectively, and increments their epoch upon receiving a message with an epoch one greater than their own, ensuring that epochs stay one step apart. This epoch-based "ping-pong" approach ensures that parties only need to exchange the hash and ordinals of messages received in the last two epochs to achieve s-RID security. Informally, a full communication round-trip implicitly acknowledges the reception and verification of the hash and ordinals. If the communication between the parties is balanced, this leads to a concrete reduction in the communication overhead.

For out-of-band UNF security, we compress authentication tags by adding explicit acknowledgements, similar to the implicit acknowledgments used in the in-band setting. Since the out-of-band channel is authentic, parties can trust the sets of sent and received messages, allowing them to prune messages that have already been authenticated.

Finally, we leverage the delayed r-UNF-like guarantee discussed above to formalize a lightweight three-move protocol and corresponding security model for bidirectional message authentication over the out-of-band channel. In the first and second moves, participant $\mathcal{P}$ (resp. $\overline{\mathcal{P}}$) sends their set of received messages to their partner. In the second and third moves, $\overline{\mathcal{P}}$ (resp. $\mathcal{P}$) sends a bit indicating whether the set of received messages is consistent with what they actually sent.

2.1.1 Summary

To summarize, in this chapter we make the following contributions:

- We introduce (Section 2.2) a new primitive, *authenticated ratcheted communication*, which captures immediate decryption and models communication through both insecure in-band and authentic out-of-band channels.
- In Section 2.3, we formalise in-band active attack detection for immediate decryption with two notions: r-RID and s-RID security. These notions capture detection of active attacks towards the receiver and on reception of messages from the sender after that the sender was attacked, respectively. Together, these notions comprise

RID security. We propose a scheme secure under these notions.

- Section 2.4 formalizes *out-of-band* active attack detection by introducing r-UNF and s-UNF notions, which together form UNF-security, analogous to the in-band notions. We construct an UNF-secure scheme from a RID-secure scheme and present an UNF-secure ARC scheme given an RC scheme.

- In Section 2.5, we explore optimization techniques. First, we show that s-RID provide r-RID-like security after a honest round trip. We then discuss ways to optimize the s-RID-secure scheme by reducing detection overhead when communication is balanced between parties. Additionally, we optimize the *authenticated* out-of-band channel variant using pruning-based techniques. Finally, we highlight the performance benefits of a three-move authentication procedure.

2.2 (Authenticated) ratcheted communication

In this section we introduce the *ratcheted communication* (RC) cryptographic primitive and an extension *authenticated ratcheted communication* (ARC) supporting out-of-band authentication. These primitives augment classic ratcheted secure messaging schemes [JMM19, ACD19, CDV21] in two ways: (i) messages that parties send and receive are associated with *ordinals*, and, for ARC, (ii) the syntax encompasses two additional stateful algorithms AuthSend and AuthReceive.

Ordinals associated with messages serve three purposes: (1) ordering incoming messages in the immediate decryption setting; (2) tracking the number of messages that have passed through the communication channel; and (3) identifying which messages have been authenticated using the out-of-band channel. Ordinals, which we denote as num, can be elements of any set on which a total order is defined. We assume that $\perp$ is always the smallest num, regardless of the set to which num belongs. In the models of Alwen et al. [ACD19] and Bienstock et al. [BFG+22a] for the Signal protocol, an ordinal num is a pair of integers (e, c) such that $(e, c) < (e', c')$ if $e < e'$ or both $e = e'$ and $c < c'$. We refer to this modeling as Signal's epoch/index pairs. Below, we formally define a ratcheted communication scheme.

Definition 1 (Ratcheted communication (RC))**.** A *ratcheted communication* (RC) scheme comprises the following four efficient algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. Given a security parameter λ, expressed in unary, return public parameters pp.

- $\mathsf{Init}(\mathsf{pp}) \to (\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z)$. Given public parameters pp return a state $\mathsf{st}_\mathcal{P}$ for $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ and public information z.

- $\mathsf{Send}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}, \mathsf{pt}) \to (\mathsf{st}'_{\mathcal{P}}, \mathsf{num}, \mathsf{ct})$. Take as input a state $\mathsf{st}_{\mathcal{P}}$, associated data ad and a plaintext pt and output a new state $\mathsf{st}'_{\mathcal{P}}$, an ordinal num and ciphertext ct.
- $\mathsf{Receive}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}, \mathsf{ct}) \to (\mathsf{acc}, \mathsf{st}'_{\mathcal{P}}, \mathsf{num}, \mathsf{pt})$. Take as input a state $\mathsf{st}_{\mathcal{P}}$, associated data ad and ciphertext ct and outputs an acceptance bit $\mathsf{acc} \in \{\mathsf{true}, \mathsf{false}\}$, state $\mathsf{st}'_{\mathcal{P}}$, ordinal num and plaintext pt.

The Receive algorithm returns dummy $\mathsf{st}'_{\mathcal{P}}$, num, pt which are ignored when $\mathsf{acc} = \mathsf{false}$.

Signal's Double Ratchet [PM16] can be viewed as an RC. In the work of Alwen et al. [ACD19], a secure messaging scheme consists of an initialisation algorithm and party-specific Send and Receive algorithms with no associated data. The Receive algorithms, but not the Send algorithms, output an epoch/index pair $(t, i) \in \mathbb{N}^2$ which plays the role of an ordinal. Signal as defined by Alwen et al. [ACD19] can thus be considered an RC by modifying its Send algorithm to output each (t, i) pair as an ordinal and enforcing that $\mathsf{ad} = \perp$ is always input to Send and Receive.

In an ARC, parties use AuthSend and AuthReceive to authenticate the communication over a possibly narrowband out-of-band authenticated channel. AuthSend takes the current state as input and outputs an updated state, an authentication tag and an ordinal. AuthReceive takes a state and an authentication tag to output an authentication bit, an updated state and an ordinal. Intuitively, the authentication that the sender sends via the out-of-band channel, enables the receiver to detect active attacks using the AuthReceive algorithm. Participants can decide when to invoke the algorithms and thus use the authentication tag on-demand, e.g., when the participants meet in person and the out-of-band channel is available.

AuthSend and AuthReceive output ordinals with the same semantics as Send and Receive. The num that AuthSend outputs is greater or equal to the last num that Send outputs; besides ordering authentication tags with respect to messages the party has sent or received, the ordinal indicates which messages (all up until num) the authentication tag authenticates. Similarly, for AuthReceive, the ordinal num indicates that the received authentication tag authenticates all messages with $\mathsf{num}' \leq \mathsf{num}$.

Definition 2 (Authenticated ratcheted communication (ARC))**.** An *authenticated ratcheted communication* (ARC) scheme comprises the following efficient algorithms:

- Setup, Init, Send, Receive defined as in RC (Definition 1).
- $\mathsf{AuthSend}(\mathsf{st}_{\mathcal{P}}) \to (\mathsf{st}'_{\mathcal{P}}, \mathsf{num}, \mathsf{at})$. Input a state $\mathsf{st}_{\mathcal{P}}$ and return a new state $\mathsf{st}'_{\mathcal{P}}$, an ordinal num and an authentication tag at.
- $\mathsf{AuthReceive}(\mathsf{st}_{\mathcal{P}}, \mathsf{at}) \to (\mathsf{auth}, \mathsf{st}'_{\mathcal{P}}, \mathsf{num})$. Take as input a state $\mathsf{st}_{\mathcal{P}}$ and authentication tag at and output an authentication bit $\mathsf{auth} \in \{\mathsf{true}, \mathsf{false}\}$, an updated state $\mathsf{st}'_{\mathcal{P}}$ and an ordinal num.

The AuthReceive algorithm returns dummy $\mathsf{st}'_{\mathcal{P}}$, num which the scheme ignores when auth = false.

One could alternatively define AuthSend/AuthReceive to output sets of ordinals, rather than single ones, corresponding to which messages have been authenticated. Our security notions ensure that this information can be efficiently computed by parties using the ordinal that the algorithms output.

We define *correctness* for an RC and ARC scheme using the CORRECT, presented in Figure 2.1. The game inputs a security parameter and a schedule sched, which models the message flow between the participants. Participants can (1) send a message, (2) receive a message, and for ARC schemes only, (3) send an authentication tag, or (4) receive a sent authentication tag. More precisely, sched is an ordered list of instructions either of the form $(\mathcal{P}, \text{“send”}, \mathsf{ad}, \mathsf{pt})$, $(\mathcal{P}, \text{“rec”}, j)$, and for ARC only, $(\mathcal{P}, \text{“authsend”})$, and $(\mathcal{P}, \text{“authrec”}, j)$, where $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$, ad denotes associated data, pt denotes a plaintext, and $j \in \mathbb{N}$ indicates either the $(\mathsf{ad}, \mathsf{ct})$ pair or the at to be received and processed by Receive or AuthReceive, respectively.

A correct (A)RC scheme must recover the correct plaintext from the corresponding associated data/ciphertext pair. Moreover, the scheme must satisfy the following properties.

- Subsequent calls to the Send algorithm outputs increasing ordinals (line 6 in Figure 2.1).[1]
- Ordinals are equal for corresponding calls to Send (resp. AuthSend for ARC) and Receive (resp. AuthReceive for ARC) (lines 11 and 21).
- For ARC, AuthSend outputs an ordinal greater or equal to the ordinal returned by the last call to Send (line 15).

We require that these properties hold in the adversarial setting (in particular when forgeries are received) and enforce them in the ORDINALS game presented in Figure 2.3.

We formally define correctness for an (A)RC scheme in Definition 3.

Definition 3 (CORRECT)**.** Consider the correctness game CORRECT presented in Figure 2.1. An RC (resp. ARC) scheme is *correct* if, for all $\lambda \in \mathbb{N}$, and all sequences of the form sched with elements of the form $(\mathcal{P}, \text{“send”}, \mathsf{ad}, \mathsf{pt})$, $(\mathcal{P}, \text{“rec”}, j)$, (resp. also of the form $(\mathcal{P}, \text{“authsend”})$, $(\mathcal{P}, \text{“authrec”}, j)$), for $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$, we have

$$\Pr[\mathsf{CORRECT}(1^\lambda, \mathsf{sched}) \Rightarrow 1] = 0.$$

[1] This could alternatively require Send to output strictly increasing ordinals based on $\mathcal{P}$'s calls to Send *and* Receive. While the work of Alwen et al. [ACD19] fulfills this requirement, we chose not to adopt it to maintain generality.

Game $\mathsf{CORRECT}(1^\lambda, \mathsf{sched})$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{st_A}, \mathsf{st_B}, z) \leftarrow \mathsf{Init}(\mathsf{pp})$
2 : $\mathsf{ad}_*[\cdot], \mathsf{pt}_*[\cdot], \mathsf{ct}_*[\cdot], \mathsf{at}_*[\cdot] \leftarrow \bot$; $\mathsf{received}[\cdot], \mathsf{sent}[\cdot] \leftarrow \mathsf{false}$; $\mathsf{sent\text{-}num}_* \leftarrow 0$
3 : **for** $i = 1$ **to** $\mathsf{length}(\mathsf{sched})$ **do**
4 : **if** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“send”}, \mathsf{ad}, \mathsf{pt})$ for $\mathsf{ad}, \mathsf{pt}, \mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
5 : $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{ct}) \leftarrow \mathsf{Send}(\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{pt})$
6 : **if** $i > 1 \wedge \mathsf{num} \leq \mathsf{sent\text{-}num}_\mathcal{P}$ **then return** 1
7 : $\mathsf{sent}[i] \leftarrow \mathsf{true}$; $\mathsf{ad}_\mathcal{P}[i] \leftarrow \mathsf{ad}$; $\mathsf{pt}_\mathcal{P}[i] \leftarrow (\mathsf{num}, \mathsf{pt})$; $\mathsf{ct}_\mathcal{P}[i] \leftarrow \mathsf{ct}$; $\mathsf{sent\text{-}num}_\mathcal{P} \leftarrow \mathsf{num}$
8 : **elseif** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“rec”}, j)$ for $j \in \mathbb{N}, \mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
9 : **if** $\neg\mathsf{sent}[j] \vee \mathsf{received}[j] \vee \mathsf{at}_{\overline{\mathcal{P}}}[j] \neq \bot$ **then continue**
10 : $(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{Receive}(\mathsf{st}_\mathcal{P}, \mathsf{ad}_{\overline{\mathcal{P}}}[j], \mathsf{ct}_{\overline{\mathcal{P}}}[j])$
11 : **if** $\neg\mathsf{acc} \vee ((\mathsf{num}, \mathsf{pt}) \neq \mathsf{pt}_{\overline{\mathcal{P}}}[j])$ **then return** 1
12 : $\mathsf{received}[j] \leftarrow \mathsf{acc}$; $\mathsf{st}_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
13 : **elseif** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“authsend”})$ for $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
14 : $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{at}) \leftarrow \mathsf{AuthSend}(\mathsf{st}_\mathcal{P})$
15 : **if** $\mathsf{num} < \mathsf{sent\text{-}num}_\mathcal{P}$ **then return** 1
16 : $\mathsf{sent}[i] \leftarrow \mathsf{true}$; $\mathsf{at}_\mathcal{P}[i] \leftarrow (\mathsf{num}, \mathsf{at})$
17 : **elseif** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“authrec”}, j)$ for $j \in \mathbb{N}, \mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
18 : **if** $\neg\mathsf{sent}[j] \vee \mathsf{received}[j] \vee \mathsf{at}_{\overline{\mathcal{P}}}[j] = \bot$ **then continue**
19 : $(\mathsf{num}_{\overline{\mathcal{P}}}, \mathsf{at}_{\overline{\mathcal{P}}}) \leftarrow \mathsf{at}_{\overline{\mathcal{P}}}[j]$
20 : $(\mathsf{auth}, \mathsf{st}'_\mathcal{P}, \mathsf{num}) \leftarrow \mathsf{AuthReceive}(\mathsf{st}_\mathcal{P}, \mathsf{at}_{\overline{\mathcal{P}}})$
21 : **if** $\neg\mathsf{auth} \vee \mathsf{num} \neq \mathsf{num}_{\overline{\mathcal{P}}}$ **then return** 1
22 : $\mathsf{received}[j] \leftarrow \mathsf{true}$; $\mathsf{st}_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
23 : **return** 0

Figure 2.1: Correctness game for an RC/ARC scheme. Highlighted statements are only executed for when considering an ARC.

Correctness states that AuthSend must output an ordinal greater or equal to the ordinal that the last call to Send returned. If AuthSend does not increase the ordinal, then it is clear which messages are authenticated; if the ordinal increases in AuthSend, the application designer must keep track of the last num that Send returned to infer what the tag authenticates. Nonetheless, the latter case may be desirable to ensure that all ordinals output by Send and AuthSend are distinct.

Our security notions for RC and ARC schemes build on a common set of oracles, introduced in Figure 2.2. The SEND (resp. RECEIVE) oracle enables the adversary to send (resp. receive) a message on behalf of a party $\mathcal{P}$. In SEND, the caller can specify the randomness used by Send with the r variable, or let the challenger sample randomness uniformly by passing $\mathsf{r} = \varepsilon$. The SEND oracle does not keep track of randomness manipulation because our security analysis do not need this information. For ARC, AUTHSEND enables the

adversary to send an authentication tag on behalf of a party $\mathcal{P}$, whereas AUTHRECEIVE handles AuthReceive. The oracles $\mathsf{EXP}_{\mathsf{pt}}(j)$ and $\mathsf{EXP}_{\mathsf{st}}(j)$ expose plaintexts and states, respectively.

All the oracles use associative arrays to store information about the calls. The state and plaintext arrays store the internal state of a party and the sent or received plaintext, respectively; they serve the exposure oracles. The log array keeps track of which operations the adversary performs and we use it to define the security properties.

Oracle SEND($\mathcal{P}$, ad, pt, r)

1 : $i \leftarrow i + 1$
2 : **if** r $= \varepsilon$ **then** r $\leftarrow\!\!\$\ \mathcal{R}$
3 : $(\mathsf{st}_{\mathcal{P}}, \mathsf{num}, \mathsf{ct}) \leftarrow \mathsf{Send}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}, \mathsf{pt}; \mathsf{r})$
4 : $\mathsf{state}[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
5 : $\mathsf{plaintext}[i] \leftarrow \mathsf{pt}$
6 : $\log[i] \leftarrow (\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct})$
7 : **return** (num, ct)

Oracle RECEIVE($\mathcal{P}$, ad, ct)

1 : $(\mathsf{acc}, \mathsf{st}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{Receive}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}, \mathsf{ct})$
2 : **if** $\neg$acc **then return** $\perp$
3 : $i \leftarrow i + 1$
4 : $\mathsf{st}_{\mathcal{P}} \leftarrow \mathsf{st};\ \mathsf{state}[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
5 : $\mathsf{plaintext}[i] \leftarrow \mathsf{pt}$
6 : $\log[i] \leftarrow (\text{“rec”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct})$
7 : **return** num

Oracle AUTHSEND($\mathcal{P}$)

1 : $i \leftarrow i + 1$
2 : $(\mathsf{st}_{\mathcal{P}}, \mathsf{num}, \mathsf{at}) \leftarrow \mathsf{AuthSend}(\mathsf{st}_{\mathcal{P}})$
3 : $\mathsf{auth}[(\mathcal{P}, i)] \leftarrow \mathsf{at}$
4 : $\mathsf{state}[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
5 : $\log[i] \leftarrow (\text{“authsend”}, \mathcal{P}, \mathsf{num}, \mathsf{at})$
6 : **return** (num, at)

Oracle AUTHRECEIVE($\mathcal{P}, j$)

1 : $\mathsf{at} \leftarrow \mathsf{auth}[(\overline{\mathcal{P}}, j)]$
2 : **if** at $= \perp$ **then return** $\perp$
3 : $(\mathsf{auth}, \mathsf{st}, \mathsf{num}) \leftarrow \mathsf{AuthReceive}(\mathsf{st}_{\mathcal{P}}, \mathsf{at})$
4 : **if** $\neg$auth **then return** $\perp$
5 : $i \leftarrow i + 1$
6 : $\mathsf{st}_{\mathcal{P}} \leftarrow \mathsf{st};\ \mathsf{state}[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
7 : $\log[i] \leftarrow (\text{“authrec”}, \mathcal{P}, \mathsf{num}, \mathsf{at})$
8 : **return** num

Oracle $\mathsf{EXP}_{\mathsf{pt}}(j)$

1 : $i \leftarrow i + 1$
2 : $\log[i] \leftarrow (\text{“ptexp”}, j)$
3 : **return** $\mathsf{plaintext}[j]$

Oracle $\mathsf{EXP}_{\mathsf{st}}(j)$

1 : $i \leftarrow i + 1$
2 : $\log[i] \leftarrow (\text{“stexp”}, j)$
3 : **return** $\mathsf{state}[j]$

Figure 2.2: Oracles which use variables state, plaintext, log, auth, st_* and i, all initialized in games where the oracles are used. AUTHSEND and AUTHRECEIVE are only used when considering ARC

The oracles of Figure 2.2 models a communication network composed of insecure in-band and authentic out-of-band channels. The SEND and RECEIVE oracles enable the adversary to read, deliver, modify and delay messages, but AUTHSEND and AUTHRECEIVE do not allow the modification of authentication tags.

We assume an *always-authentic* out-of-band channel. To our knowledge, all deployed solution for out-of-band authentication and relevant literature [DH20, DGP22] assume this. One can define a stronger model where the out-of-band channel is authentic only in some cases, e.g., the tampering rate is bounded, or multiple out-of-band channels exist but the adversary can compromise only a subset of them. As a not-always-authentic out-of-band channel is a stronger version of an insecure in-band channel, the discussions in Section 2.3 apply.

For RC and ARC schemes we require that even in the presence of an adversary that injects forgeries, the Send and Receive (as well as AuthSend and AuthReceive for ARC schemes) algorithms output correct ordinals. An RC or ARC scheme has *correct ordinals* if (1) the Send algorithm always outputs increasing ordinals with respect to all previously sent or received ordinals; (2) corresponding calls to Send and Receive (resp. to AuthSend and AuthReceive) output the same ordinal; and (3) for an ARC scheme, AuthSend outputs an ordinal greater or equal to the ordinal returned by the last call to Send. We consider these properties in CORRECT (Figure 2.1), but they must hold also in presence of forgeries. We formalize this notion with the ORDINALS game in Figure 2.3.

In this game the challenger verifies three predicates, which correspond to the conditions for correct ordinals presented above. In Definition 4 we formalize ORDINALS-security for (A)RC schemes.

Definition 4 (ORDINALS)**.** Consider the ORDINALS game in Figure 2.3. We say that an (authenticated) ratcheted communication scheme is ORDINALS secure if, for all possibly unbounded adversaries $\mathcal{A}$ we have

$$\Pr[\mathsf{ORDINALS}^{\mathcal{A}}(1^{\lambda}) \Rightarrow 1] = 0.$$

The ORDINALS game in Figure 2.3 is not suited to the case where ordinals can be arbitrary and in particular collide between parties. Thus, the protocol must associate each party with disjoint ordinals: practical schemes like the Signal protocol do this by associating one party with even epochs and the counterpart with odd epochs.

2.3 In-band active attack detection: RID

In this section we consider *in-band* active attack detection in the immediate decryption setting.

Caforio et al. [CDV21] define RECOVER security, which encompasses both r-RECOVER security and s-RECOVER security, by assuming that the channel ensures in-order message delivery. Intuitively, r-RECOVER security prevents a party from being able to deliver an honest message *after* delivering a forgery, and s-RECOVER security allows a party

Game $\mathsf{ORDINALS}^{\mathcal{A}}(1^\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{st_A}, \mathsf{st_B}, z) \leftarrow \mathsf{Init}(\mathsf{pp})$
$\mathsf{state}[\cdot], \mathsf{plaintext}[\cdot], \log[\cdot], \mathsf{auth}[\cdot], \mathsf{st}_* \leftarrow \bot$
$i \leftarrow 0$
$\mathcal{A}^{\mathsf{SEND},\mathsf{RECEIVE},\mathsf{EXP_{pt}},\mathsf{EXP_{st}},\mathsf{AUTHSEND},\mathsf{AUTHRECEIVE}}(z)$
if $\exists\ \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}, \mathsf{ct}, x, y$:
  $\mathsf{not\text{-}increasing}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', x, y) \vee$
  $\mathsf{different}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}, \mathsf{ct}, \mathsf{at}) \vee$
  $\mathsf{auth\text{-}monotonic}(\log, \mathcal{P}, \mathsf{num}', y)$ **then**
    return 1
return 0

$\mathsf{not\text{-}increasing}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', x, y)$

return $((\text{“}\mathsf{send}\text{”}, \mathcal{P}, \mathsf{num}, \cdot, \cdot) = \log[x] \vee (\text{“}\mathsf{rec}\text{”}, \mathcal{P}, \mathsf{num}, \cdot, \cdot) = \log[x]) \wedge$
  $(\text{“}\mathsf{send}\text{”}, \mathcal{P}, \mathsf{num}', \cdot, \cdot) = \log[y] \wedge (0 < x < y) \wedge (\mathsf{num} \geq \mathsf{num}')$

$\mathsf{different}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}, \mathsf{ct}, \mathsf{at})$

return $(((\text{“}\mathsf{send}\text{”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \in \log \wedge (\text{“}\mathsf{rec}\text{”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{ad}, \mathsf{ct}) \in \log)) \vee$
  $((\text{“}\mathsf{authsend}\text{”}, \mathcal{P}, \mathsf{num}, \mathsf{at}) \in \log \wedge (\text{“}\mathsf{authrec}\text{”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{at}) \in \log))) \wedge (\mathsf{num} \neq \mathsf{num}')$

$\mathsf{auth\text{-}monotonic}(\log, \mathcal{P}, \mathsf{num}', y)$

$\mathsf{num} \leftarrow \max\{\bot \cup \{\mathsf{num}'' : (\text{“}\mathsf{send}\text{”}, \mathcal{P}, \mathsf{num}'', \cdot, \cdot) = \log[x] \wedge 0 < x < y\}\}$
return $(\text{“}\mathsf{authsend}\text{”}, \mathcal{P}, \mathsf{num}, \cdot, \cdot) = \log[y] \wedge (\mathsf{num} > \mathsf{num}')$

Figure 2.3: ORDINALS game. Highlighted statements are only considered for an ARC.

to detect and stop communication when their partner has delivered a forgery. We extend these notions to handle out-of-order message delivery by introducing r-RID and s-RID, which we formally present in Figure 2.4 and illustrate in Figure 2.5. Combined, these two properties ensure RID security. Note that these definitions are orthogonal to the usual forward and post-compromise security notions that the ratcheting literature considers [BSJ+17, ACD19].

The winning condition in RID consists of three predicates:

- forgery verifies whether a forgery was accepted by one of the participants by taking into account both injection and modification of messages. In the predicate, we denote the impersonated party as $\mathcal{P}$ and the recipient of the forgery as $\overline{\mathcal{P}}$.
- bad-$\overline{\mathsf{P}}$ checks whether the recipient of the forgery manages to detect the attack. This predicate corresponds to r-RID security.

- bad-P establishes whether $\mathcal{P}$, i.e., the participant that the adversary impersonates to send the forgery, fails to detect the attack. Since $\overline{\mathcal{P}}$ is the recipient of the forgery, the detection of the attack by $\mathcal{P}$ relies on a ciphertext sent by $\overline{\mathcal{P}}$ and honestly delivered. This predicate corresponds to s-RID security.

The game imposes that if forgery returns true, then at least one between bad-P and bad-$\overline{\text{P}}$ must return true for the adversary to win the game.

Definition 5 (RID)**.** A RC is (q, t, ϵ)-r-RID (resp. s-RID) secure, if for all adversaries $\mathcal{A}$ which make at most q oracle queries and which run in time at most t, we have:

$$\Pr[\text{r-RID}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon \text{ (resp. } \Pr[\text{s-RID}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon),$$

where game r-RID$^{\mathcal{A}}$ (resp. s-RID$^{\mathcal{A}}$) is defined in Figure 2.4.

Game r-RID$^{\mathcal{A}}(1^\lambda)$ | s-RID$^{\mathcal{A}}(1^\lambda)$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{st_A}, \mathsf{st_B}, z) \leftarrow \mathsf{Init}(\mathsf{pp})$
2 : $\mathsf{state}[\cdot], \mathsf{plaintext}[\cdot], \log[\cdot] \leftarrow \bot$
3 : $\mathsf{auth}[\cdot], \mathsf{st}_* \leftarrow \bot$
4 : $i \leftarrow 0$
5 : $\mathcal{A}^{\mathcal{O}}(z)$
6 : **if** $\exists\ \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}, \mathsf{ct}, \mathsf{ad}', \mathsf{ct}', x, y$:
7 : $\quad \mathsf{forgery}(\log, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x) \wedge$
8 : $\quad \text{bad-}\overline{\text{P}}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ **then**
9 : $\quad$ bad-P$(\log, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y)$ **then**
10 : $\quad\quad$ **return** 1
11 : **return** 0

Game RID$^{\mathcal{A}}(1^\lambda)$

1 : **return** r-RID$^{\mathcal{A}}(1^\lambda) \vee$ s-RID$^{\mathcal{A}}(1^\lambda)$

forgery$(\log, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x)$

1 : **return** $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \log \wedge$
2 : $\quad (\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \log[x]$

bad-$\overline{\text{P}}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}')$

1 : **return** $(\text{“send”}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \log \wedge$
2 : $\quad (\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \log \wedge$
3 : $\quad (\mathsf{num} < \mathsf{num}')$

bad-P$(\log, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y)$

1 : **return** $(y > x) \wedge$
2 : $\quad (\text{“send”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') = \log[y] \wedge$
3 : $\quad (\text{“rec”}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \log$

Figure 2.4: r-RID, s-RID and RID games for $\mathcal{O} = \{\mathsf{SEND}, \mathsf{RECEIVE}, \mathsf{EXP_{pt}}, \mathsf{EXP_{st}}\}$.

Although r-RID seems to be stronger than s-RID at first glance, the two notions are not comparable. There exist schemes which provide r-RID and not s-RID security and vice versa, e.g., the scheme proposed in Figure 2.7 if the checks for either r-RID or s-RID are removed from the checks subroutine given the underlying RC is not r-RID or s-RID secure, respectively (the Double Ratchet is neither, for example).

However, we observe the following link between the two notions. In a s-RID-secure scheme, $\mathcal{P}$ can detect that $\overline{\mathcal{P}}$ has received a forged message. If $\mathcal{P}$ subsequently sends an “abort”

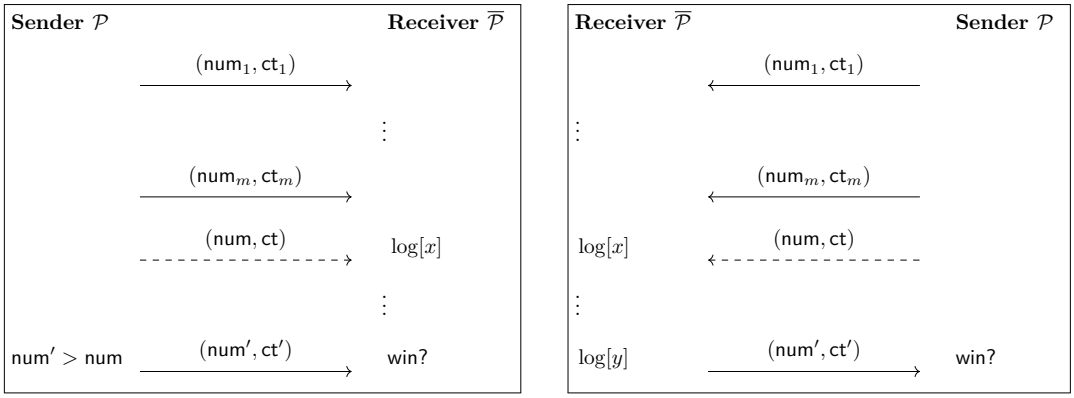

Figure 2.5: Visualizing r-RID (left) and s-RID (right). Each figure showcases an adversary's winning condition in the respective game. The dashed arrows are forged messages. If $\overline{\mathcal{P}}$ accepts the message at time "win?" then the adversary wins.

message to $\overline{\mathcal{P}}$, $\overline{\mathcal{P}}$ will be able to detect the forgery after *one* honest round-trip of messages. In other words, s-RID RC schemes can be transformed (by adding an "abort" message) into RC schemes with a weak variant of r-RID security: r-RID after a honest round-trip. We discuss this delayed guarantee further in Section 2.5.1.

Fine-grained security. Suppose A sends five messages with $\mathsf{num} \in \{1, \ldots, 5\}$, B receives a forgery with $\mathsf{num} = 1000$, and then A sends five messages with $\mathsf{num} \in \{6, \ldots, 10\}$. If B never sends (i.e., A is the sender and B the receiver), RID-security only guarantees that the forgery might be detected when A sends the honest message with $\mathsf{num}' = 1001$. Indeed, the condition "$\mathsf{num} < \mathsf{num}'$" in predicate bad-$\overline{\mathsf{P}}$ in Figure 2.4 mandates that the honest message that $\mathcal{P}$ (A in our example) sends and $\overline{\mathcal{P}}$, the victim of the forgery B in our example, correctly receives, must detect the forgery if the forgery num is smaller than the num' of the honest message. Intuitively, B should be able to detect the forgery upon receiving the honest message with $\mathsf{num} = 6$, as this message is "independent" of the forgery with $\mathsf{num} = 1000$. By the not-increasing predicate of ORDINALS security (cf. Figure 2.3), all messages that A sends after one round-trip will have $\mathsf{num} > 1000$, meaning that such an attack will be eventually detected. Capturing fine-grained security for these scenarios require tracking state exposures and message delivery timing, adding definitional complexity; we leave this as an open area for further exploration. While our construction below defeats some forgeries, it is likely necessary to leverage the security of the underlying RC to build a scheme providing this kind of fine-grained security. Looking ahead, this observation also applies to UNF-secure ARC schemes defined in Section 2.4.

Game $\mathsf{CR}^{\mathcal{A}}(1^{\lambda})$

1 : $\mathsf{hk} \leftarrow \mathsf{Gen}(1^{\lambda})$
2 : $m_1, m_2 \leftarrow \mathcal{A}(\mathsf{hk})$
3 : **return** $\mathbb{1}\{\mathsf{H.Eval}(\mathsf{hk}, m_1) = \mathsf{H.Eval}(\mathsf{hk}, m_2) \wedge m_1 \neq m_2\}$

Figure 2.6: Collision resistance game CR for a hash function H.

2.3.1 A RID-secure RC

In this section we build a RID-secure RC scheme given a correct and ORDINALS-secure RC scheme and a collision-resistant hash function H (Definition 7). We present our transformation in Figure 2.7.

Preliminary: collision resistant hash function

We formally introduce the syntax of an hash function and define collision resistance. For theoretical reasons within the concrete-security framework, we define a keyed hash function; see Rogaway's work for more details [Rog06].

Definition 6 (Hash function)**.** A hash function H comprises the following efficient algorithms:

- $\mathsf{Gen}(1^{\lambda}) \to \mathsf{hk}$. Given a security parameter λ, expressed in unary, return a hash key hk.
- $\mathsf{Eval}(\mathsf{hk}, \mathsf{pt}) \to h$. Input a hash key hk and a message $\mathsf{pt} \in \{0,1\}^*$ and output digest h.

Definition 7 (Collision resistance)**.** We say that a hash function H is (t, ϵ)-collision resistant if, for all adversaries $\mathcal{A}$ running in time at most t, we have:

$$\Pr[\mathsf{CR}^{\mathcal{A}}(1^{\lambda}) \Rightarrow 1] \leq \epsilon,$$

where game $\mathsf{CR}^{\mathcal{A}}$ is defined in Figure 2.6.

Scheme description

Each party $\mathcal{P}$ keeps track of every message it has sent and received (in S and R, respectively). This information is communicated to $\overline{\mathcal{P}}$ every time $\mathcal{P}$ send a message by calling Send (via variables S and R′).

RRC.Setup(1^λ)

1 : $\mathsf{pp}_0 \leftarrow \mathsf{RC.Setup}(1^\lambda)$
2 : $\mathsf{hk} \leftarrow \mathsf{H.KGen}(1^\lambda)$
3 : $\mathsf{hk}' \leftarrow \mathsf{H.KGen}(1^\lambda)$
4 : $\mathsf{pp} \leftarrow (\mathsf{pp}_0, \mathsf{hk}, \mathsf{hk}')$
5 : **return** pp

RRC.Send($\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{pt}$)

1 : $(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \mathsf{hk}', \mathsf{S}, \mathsf{R}, \cdot, \cdot) \leftarrow \mathsf{st}_\mathcal{P}$
2 : $\mathsf{nums}' \leftarrow \{\mathsf{num}' : (\mathsf{num}', \cdot) \in \mathsf{R}\}$
3 : $\mathsf{R}' \leftarrow (\mathsf{nums}', \mathsf{H.Eval}(\mathsf{hk}', \mathsf{R}))$
4 : $\mathsf{ad}' \leftarrow (\mathsf{ad}, \mathsf{S}, \mathsf{R}')$
5 : $(\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{ct}') \leftarrow \mathsf{RC.Send}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}', \mathsf{pt})$
6 : $\mathsf{ct} \leftarrow (\mathsf{ct}', \mathsf{S}, \mathsf{R}')$
7 : $h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$
8 : $\mathsf{st}_\mathcal{P}.\mathsf{S} \leftarrow \mathsf{S} \cup \{(\mathsf{num}, h)\}$
9 : **return** $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{ct})$

RRC.Receive($\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{ct}$)

1 : $(\mathsf{ct}', \mathsf{S}^{\overline{\mathcal{P}}}, \mathsf{R}^{\overline{\mathcal{P}}}) \leftarrow \mathsf{ct}$
2 : $(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \cdot, \cdot, \mathsf{R}, \mathsf{S}_{\mathsf{ack}}, \cdot) \leftarrow \mathsf{st}_\mathcal{P}$
3 : $\mathsf{ad}' \leftarrow (\mathsf{ad}, \mathsf{S}^{\overline{\mathcal{P}}}, \mathsf{R}^{\overline{\mathcal{P}}})$
4 : $(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{RC.Receive}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}', \mathsf{ct}')$
5 : **if** $\neg\mathsf{acc}$ **then return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
6 : $h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$
7 : **if** $\mathsf{checks}(\mathsf{st}_\mathcal{P}, \mathsf{ct}, h, \mathsf{num})$ **then**
8 : **return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
9 : $\mathsf{st}_\mathcal{P}.\mathsf{R} \leftarrow \mathsf{R} \cup \{(\mathsf{num}, h)\}$
10 : $\mathsf{st}_\mathcal{P}.\mathsf{S}_{\mathsf{ack}} \leftarrow \mathsf{S}_{\mathsf{ack}} \cup \mathsf{S}^{\overline{\mathcal{P}}}$
11 : $\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
12 : **return** $(\mathsf{acc}, \mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{pt})$

RRC.Init(pp)

1 : $(\mathsf{pp}_0, \mathsf{hk}, \mathsf{hk}') \leftarrow \mathsf{pp}$
2 : $(\mathsf{st}'_\mathsf{A}, \mathsf{st}'_\mathsf{B}, z') \leftarrow \mathsf{RC.Init}(\mathsf{pp}_0)$
3 : $\mathsf{max\text{-}num} \leftarrow \bot$
4 : $\mathsf{S}, \mathsf{R}, \mathsf{S}_{\mathsf{ack}} \leftarrow \emptyset$
5 : $\mathsf{st}_\mathsf{A} \leftarrow (\mathsf{st}'_\mathsf{A}, \mathsf{hk}, \mathsf{hk}', \mathsf{S}, \mathsf{R}, \mathsf{S}_{\mathsf{ack}}, \mathsf{max\text{-}num})$
6 : $\mathsf{st}_\mathsf{B} \leftarrow (\mathsf{st}'_\mathsf{B}, \mathsf{hk}, \mathsf{hk}', \mathsf{S}, \mathsf{R}, \mathsf{S}_{\mathsf{ack}}, \mathsf{max\text{-}num})$
7 : $z \leftarrow (z', \mathsf{pp})$
8 : **return** $(\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z)$

checks($\mathsf{st}_\mathcal{P}, \mathsf{ct}, h, \mathsf{num}$)

1 : $(\mathsf{nums}', h') \leftarrow \mathsf{ct.R}$
2 : $\mathsf{R}^* \leftarrow \{(\mathsf{num}', \cdot) \in \mathsf{st}_\mathcal{P}.\mathsf{S} : \mathsf{num}' \in \mathsf{nums}')\}$
3 : $\mathsf{s\text{-}bool} \leftarrow (\mathsf{H.Eval}(\mathsf{st}_\mathcal{P}.\mathsf{hk}', \mathsf{R}^*) \neq h')$
4 : $\mathsf{R}' \leftarrow \{(\mathsf{num}', \cdot) \in \mathsf{st}_\mathcal{P}.\mathsf{R} : \mathsf{num}' \leq \mathsf{num}\}$
5 : $\mathsf{r\text{-}bool} \leftarrow (\mathsf{R}' \not\subseteq \mathsf{ct.S})$
6 : $\mathsf{r\text{-}bool} \leftarrow \mathsf{r\text{-}bool} \vee$
7 : $(\exists(\mathsf{num}^*, \cdot) \in \mathsf{ct.S} : \mathsf{num}^* \geq \mathsf{num})$
8 : **if** $\mathsf{num} < \mathsf{st}_\mathcal{P}.\mathsf{max\text{-}num}$ **then**
9 : $\mathsf{r\text{-}bool} \leftarrow \mathsf{r\text{-}bool} \vee ((\mathsf{num}, h) \notin \mathsf{st}_\mathcal{P}.\mathsf{S}_{\mathsf{ack}})$
10 : $\mathsf{r\text{-}bool} \leftarrow \mathsf{r\text{-}bool} \vee (\mathsf{ct.S} \not\subseteq \mathsf{st}_\mathcal{P}.\mathsf{S}_{\mathsf{ack}})$
11 : $\mathsf{S_{ack}}' \leftarrow \{(\mathsf{num}', \cdot) \in \mathsf{st}_\mathcal{P}.\mathsf{S}_{\mathsf{ack}} :$
12 : $\mathsf{num}' < \mathsf{num}\}$
13 : $\mathsf{r\text{-}bool} \leftarrow \mathsf{r\text{-}bool} \vee (\mathsf{S_{ack}}' \not\subseteq \mathsf{ct.S})$
14 : **else**
15 : $\mathsf{st}_\mathcal{P}.\mathsf{max\text{-}num} \leftarrow \mathsf{num}$
16 : $\mathsf{r\text{-}bool} \leftarrow \mathsf{r\text{-}bool} \vee$
17 : $(\exists(\mathsf{num}', \cdot) \in \mathsf{st}_\mathcal{P}.\mathsf{S}_{\mathsf{ack}} \setminus \mathsf{ct.S} :$
18 : $\mathsf{num}' < \mathsf{st}_\mathcal{P}.\mathsf{max\text{-}num})$
19 : **return** $\mathsf{r\text{-}bool} \vee \mathsf{s\text{-}bool}$

Figure 2.7: RID-secure RC scheme RRC based on a RC scheme RC (Definition 1) and a hash function H (Definition 6). RRC requires the following variables: max-num represents the largest received num. S is the set of (num, h) pairs; R is the set of received (num, h) pairs; $\mathsf{S}_{\mathsf{ack}}$ is the set of (num, h) which are expected to be received (according to the received ciphertext ct). All sets are append-only.

The Send procedure prepares the set R', which contains the ordinals and a hash of all received messages (line 3). This step can be optimized by using an incremental hash function as we discuss in Section 2.5.2. Next, it calls RC.Send with input $(\mathsf{ad}', \mathsf{pt})$ where $\mathsf{ad}' = (\mathsf{ad}, \mathsf{S}, \mathsf{R}')$ is the associated data. The ciphertext ct contains both ct' and sets S and R'. Finally, it adds the pair (num, h) to S (line 8), where the hash h is computed as $\mathsf{H.Eval}(\mathsf{st}_{\mathcal{P}}.\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$, where $\mathsf{ct} = (\mathsf{ct}', \mathsf{S}, \mathsf{R}')$. Intuitively, (num, h) acts as a summary of $\mathcal{P}$'s state after calling RC.Send which can be checked by $\overline{\mathcal{P}}$ for inconsistency.

When $\overline{\mathcal{P}}$ invokes Receive, the procedure calls RC.Receive, which outputs $\mathsf{num} \neq \perp$ if the call is successful. Since ct contains $\mathsf{R}^{\overline{\mathcal{P}}}$, $\overline{\mathcal{P}}$ checks that what $\mathcal{P}$ received so far was correct (line 3 in checks). In addition, using the S set contained in the ciphertext ct, $\overline{\mathcal{P}}$ can further check whether the ciphertexts received so far have indeed been sent by $\mathcal{P}$. This is verified from lines 5 to 18 of the checks procedure. Some checks detect tampering of ct by the adversary (e.g. ct.S should not contain ordinals larger than the one of the current ciphertext, or if ct was sent earlier than another ciphertext already received, ct.S should be consistent with messages already acknowledged, etc.). If everything verifies, Receive stores (num, h) in R and adds ct.S to the set of acknowledged messages (lines 9 and 10).

The sets S and R' included in the ciphertext are also included in the authenticated data passed to the underlying RC scheme: the tuple $(\mathsf{S}, \mathsf{R}')$ is always authenticated in Figure 2.7. This is actually not needed for RID security, but for authentication and confidentiality. Even if we do not define authentication and confidentiality for (A)RC, we use authenticated encryption for completeness.

Remark 8. Our scheme outputs a generic error symbol $\perp$ in all cases. In particular, it returns the same symbol regardless of whether the error was due to detecting active attack, or from the situation where the adversary did not expose any states and simply send a malformed ciphertext. Treating both scenarios identically could lead to a denial-of-service attack vector, so in practice, they should be distinguished. We leave this differentiation to future work.

Security analysis

Correctness of the RRC scheme follows from the correctness of the underlying scheme RC and the fact that the checks always outputs false when only honest messages are received. Similarly, ORDINALS-security follows from the ORDINALS-security of RC, as RRC outputs the same num that RC outputs. As the next theorems state, the construction of Figure 2.7 is r-RID-secure (Theorem 9) and s-RID-secure (Theorem 10), and therefore RID-secure.

Theorem 9. Let H be a (t_{cr}, ϵ_{cr})-collision resistant hash function. Then RRC (defined in Figure 2.7) is a (q, t, ϵ_{cr})-r-RID-secure RC where $t_{cr} \approx t$ and q is upper bounded by t.

Proof. Let us assume there exists an adversary $\tilde{\mathcal{A}}$ playing the r-RID game, running in time $\tilde{t}$ and making at most $\tilde{q}$ queries. We call the advantage of this adversary $\tilde{\epsilon}$, hence we have

$$\Pr[\mathsf{r\text{-}RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1] = \tilde{\epsilon}.$$

Let E be an event that occurs when $\mathsf{r\text{-}RID}_{\mathsf{RRC}}(\tilde{\mathcal{A}})$ outputs 1. The strategy is to construct an adversary $\mathcal{A}^*$, running in time $\approx \tilde{t}$ such that

$$\Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 \mid E] = 1.$$

By definition of r-RID, E occurring means there exist $\mathcal{P}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}), (\mathsf{num}', \mathsf{ad}', \mathsf{ct}'), x$ such that $\mathsf{bad\text{-}\overline{P}}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ and $\mathsf{forgery}(\log, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x)$ are both true. This means that the message with ordinal num was not sent by $\mathcal{P}$ but received at some point by $\overline{\mathcal{P}}$ (cf. the condition $(\text{“}\mathsf{rec}\text{”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \log[x]$ in the $\mathsf{forgery}$ predicate), and the message with ordinal num', with $\mathsf{num} < \mathsf{num}'$, was also received and was actually sent by $\mathcal{P}$ (cf. the $\mathsf{bad\text{-}\overline{P}}$ predicate).

We separate the two cases where 1) the message with ordinal num (the forged message) is received before the message with ordinal num' (the honest message), and 2) the message with ordinal num' is received first. We first analyse the former case.

We argue that unless the adversary found a collision, the message with ordinal num' (the honest message) would not have been delivered. Assume by contradiction that the honest message *was* successfully delivered. Let $\mathsf{st}_{\mathcal{P}}$ be the state of the receiver $\mathcal{P}$ while receiving message num', and $\mathsf{st}_{\overline{\mathcal{P}}}$ be the state of the sender $\overline{\mathcal{P}}$ while sending the message $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$. As $(\text{“}\mathsf{rec}\text{”}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \log$, it means $\mathsf{RRC.Receive}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}', \mathsf{ct}') \rightarrow (\mathsf{true}, \mathsf{st}'_{\mathcal{P}}, \mathsf{num}', \mathsf{pt}')$, which implies that $\mathsf{checks}(\mathsf{st}_{\mathcal{P}}, \mathsf{ct}', \mathsf{num}', \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}', \mathsf{ad}', \mathsf{ct}')))$ returned **false**.

As $\mathsf{num} \leq \mathsf{num}'$, we have

$$(\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))) \in \mathsf{R}' \quad \text{and} \quad (\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))) \in \mathsf{ct}'.\mathsf{S},$$

as otherwise r-bool would have been set to true in line 5. Let $h_f \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$. As $(\mathsf{num}, h_f) \in \mathsf{ct}'.\mathsf{S}$, we have

$$(\mathsf{num}, h_f) \in \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S} \tag{2.1}$$

since ct' was sent by $\overline{\mathcal{P}}$. This would mean that $\overline{\mathcal{P}}$ did send a message with ordinal num, let us call it $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)$. Hence, we have that,

$$(\mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}_h, \mathsf{ct}_h, \mathsf{num})), \mathsf{num}) \in \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S} \tag{2.2}$$

By combining (2.1) and (2.2) and the fact that num can appear only once in $\mathsf{st}_{\overline{\mathcal{P}}}$ (since

RRC is ORDINALS-secure), we get the collision

$$\mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)) = \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct})).$$

This is a collision since $(\mathsf{ad}_h, \mathsf{ct}_h) \neq (\mathsf{ad}, \mathsf{ct})$ as $(\text{“}\mathsf{send}\text{”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \in \log$ and $(\text{“}\mathsf{send}\text{”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \log$.

Now consider the case where $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ is received before $(\mathsf{num}, \mathsf{ad}, \mathsf{ct})$. Since $\mathsf{num} \leq \mathsf{num}'$, this implies that, while receiving $(\mathsf{num}, \mathsf{ad}, \mathsf{ct})$, $\mathsf{max\text{-}num} \geq \mathsf{num}' \geq \mathsf{num}$. Consequently, $(\mathsf{num}, h) \in \mathsf{st}_{\mathcal{P}}.\mathsf{S}_{\mathsf{ack}}$, otherwise the condition on line 9 would not have been satisfied. Since $\mathsf{S}_{\mathsf{ack}}$ is updated only by adding the elements in $\mathsf{S}^{\overline{\mathcal{P}}}$ when a message is received, and since (num, h_f) is not in $\mathsf{ct}^*.\mathsf{S}$, for any honest ct^*, there must exist a forged message $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ received before $(\mathsf{ad}, \mathsf{ct}, \mathsf{num})$ such that $(\mathsf{num}, h_f,) \in \mathsf{ct}''.\mathsf{S}$. Given that we considered $(\mathsf{num}, \mathsf{num}')$ as the first pair of messages violating the r-RID property, it follows that $\mathsf{num}'' > \mathsf{num}' > \mathsf{num}$.

We consider two cases: when $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ is received before $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$, and when it is received afterward. Let us consider the first case. In this case, we argue that the honest message $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ would not be accepted. Since num'' is received before num', it follows that $\mathsf{num}' < \mathsf{num}'' \leq \mathsf{max\text{-}num}$. Moreover, since $\mathsf{r\text{-}bool} = \mathsf{false}$, it holds that $\mathsf{S}'_{\mathsf{ack}} \subseteq \mathsf{ct}'.\mathsf{S}$ (line 12). However $(\mathsf{num}, h_f) \in \mathsf{S}'_{\mathsf{ack}}$, as it was in $\mathsf{ct}''.\mathsf{S}$, hence it should also be in $\mathsf{ct}'.\mathsf{S}$. This implies that $h_f = \mathsf{H.Eval}(\mathsf{hk}, \mathsf{ad}_h, \mathsf{ct}_h, \mathsf{num})$, leading once again to a collision.

Now let us consider the case where the message $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ is received after the message $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$. We argue that $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ should not have been accepted. We split the cases where $\mathsf{num}'' \geq \mathsf{max\text{-}num}$ and $\mathsf{num}'' < \mathsf{max\text{-}num}$. Let us consider the later first. As $(\mathsf{ad}'', \mathsf{ct}'', \mathsf{num}'')$ was accepted, and hence $\mathsf{r\text{-}bool} = \mathsf{false}$, $\mathsf{ct}''.\mathsf{S} \subset \mathsf{S}_{\mathsf{ack}}$ (line 10). Now $(\mathsf{num}, h_f) \in \mathsf{ct}''.\mathsf{S}$, so $(\mathsf{num}, h_f) \in \mathsf{S}_{\mathsf{ack}}$. As without loss of generality we can imagine $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ being the first message vouching for (h_f, num), this would mean (num, h_f) was added to $\mathsf{S}_{\mathsf{ack}}$ by an honest message, i.e. $h_f = h_h$ which leads to a collision again.

Finally for the case in which $\mathsf{num}'' \geq \mathsf{max\text{-}num}$, again, considering that $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ is the first message vouching for (h_f, num), we have $(\mathsf{num}, h_f) \in \mathsf{S}_{\mathsf{ack}} \setminus \mathsf{ct}''.\mathsf{S}$ (and so r-bool would be set to true) unless $h_f = h_h$. Moreover, at this point $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ has already been received so, $\mathsf{max\text{-}num} \geq \mathsf{num}' > \mathsf{num}$. Hence, unless $h_f = h_h$, r-bool would be set to true in line 7. This concludes the proof that $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$, $(\mathsf{num}, \mathsf{ad}, \mathsf{ct})$ are accepted if and only if $\mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)) = \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$.

Now we describe the CR adversary $\mathcal{A}^*$. $\mathcal{A}^*$ runs the initialisation of RRC by replacing the sampling step of hk with the hk given by the CR game, then runs $\tilde{\mathcal{A}}$ as a subroutine, and computes $(\text{“}\mathsf{rec}\text{”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}_f, \mathsf{ct}_f) \in \log$, and $(\text{“}\mathsf{send}\text{”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \in \log$ such that $(\mathsf{ad}_f, \mathsf{ct}_f) \neq (\mathsf{ad}_h, \mathsf{ct}_h)$ and $h_f = h_h$ if possible. Given event E, this pair always exists as we have $\Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 \mid E] = 1$. Moreover, as $\mathcal{A}^*$ is just running $\tilde{\mathcal{A}}$ as a subroutine

and not doing anything extra, the time it runs is also $\approx \tilde{t}$. Finally, we have

$$\Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1] \geq \Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 | E] \cdot \Pr[E] = \Pr[\mathsf{r\text{-}RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1] = \tilde{\epsilon}.$$

Hence, $\tilde{\epsilon} \leq \Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1]$. This means that if H is (t_{cr}, ϵ_{cr})-collision resistant, then the RRC scheme is (q, t, ϵ_{cr})-$\mathsf{r\text{-}RID}$ secure with $t_{cr} \approx t$. □

Theorem 10. Let H be a (t_{cr}, ϵ_{cr})-collision resistant hash function. Then RRC (defined in Figure 2.7) is a (q, t, ϵ_{cr})-$\mathsf{s\text{-}RID}$-secure RC where $t_{cr} \approx t$ and q is upper bounded by t.

The $\mathsf{s\text{-}RID}$ security of RRC reduces to the collision resistance of the hash function H. The proof strategy is identical to the one that we use for the proof of Theorem 9, we therefore present the complete proof in Appendix A.1.

Optimizations

The $\mathsf{s\text{-}RID}$ notion imposes less overhead than $\mathsf{r\text{-}RID}$, and the construction can be further optimized while retaining $\mathsf{s\text{-}RID}$ security. Section 2.5 covers our optimizations, with Section 2.5.2 discussing why $\mathsf{s\text{-}RID}$ incurs less overhead than $\mathsf{r\text{-}RID}$, and Section 2.5.3 presenting an optimization for $\mathsf{s\text{-}RID}$ that leverages communication epochs—such as Signal's epoch/index pairs [ACD19]—to reduce the communication overhead associated with this notion.

2.4 Out-of-band active attack detection: UNF

In-band active attack detection is not always possible, as an adversary can block all honest messages within the network. For example, modern messaging solutions often rely on a possibly malicious third party server to relay messages between participants. This introduces a single point of failure for in-band communication. Consequently, we consider out-of-band active attack detection, where parties exchange authentication tags through an authenticated out-of-band channel, and define unforgeable security (UNF).

An ARC scheme is *unforgeable* if, as soon as one of the two parties accepts a forgery, both parties can detect this out-of-band. We formalize this security notion through the UNF game (Figure 2.8), which, similarly to RID, encompasses $\mathsf{r\text{-}UNF}$ and $\mathsf{s\text{-}UNF}$. The winning condition in UNF consists of three predicates: $\mathsf{forgery}, \mathsf{bad\text{-}\overline{P}}$ (corresponding to $\mathsf{r\text{-}UNF}$) and $\mathsf{bad\text{-}P}$ (corresponding to $\mathsf{s\text{-}UNF}$) that are essentially the same as the predicates that we use to define RID security (Definition 5), except they rely on authentication tags instead of ciphertexts for forgery detection. This also implies that parties perform active attack detection on-demand, i.e., only when they use the out-of-band channel.

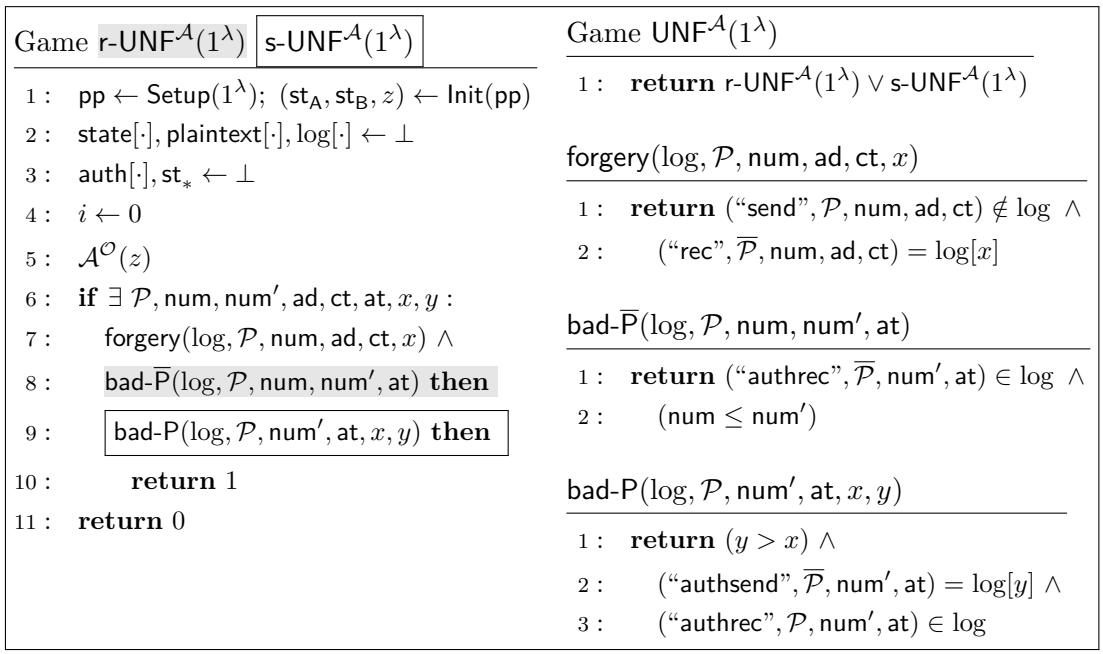

Figure 2.8: r-UNF, s-UNF and UNF games for $\mathcal{O} = \{\mathsf{SEND}, \mathsf{RECEIVE}, \mathsf{EXP_{pt}}, \mathsf{EXP_{st}}, \mathsf{AUTHSEND}, \mathsf{AUTHRECEIVE}\}$.

Definition 11 (UNF)**.** Consider the r-UNF (resp. s-UNF) game in Figure 2.8. We say that an ARC scheme is (q, t, ϵ)-r-UNF (resp. s-UNF) secure if, for all adversaries $\mathcal{A}$ which make at most q oracle queries, and which run in time at most t, we have:

$$\Pr[\text{r-UNF}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon \text{ (resp. } \Pr[\text{s-UNF}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon).$$

As for RC schemes, we do not define message indistinguishability for ARC schemes. All the schemes include in the authentication tag only *public* material, i.e., messages that have already transited through the insecure channel. Since the adversary already has access to the entire transcript of the insecure channel, the authentication material does not give any additional advantage in a message indistinguishability game.

2.4.1 UNF-secure ARC from a RID-secure RC

This section shows that RID-secure RC schemes imply UNF-secure ARC schemes. Specifically, we construct an UNF-secure ARC scheme from a RID-secure RC (see Figure 2.9). The ARC scheme uses the Setup, Gen, Init, Send, Receive functions of the underlying scheme RC. To send an authentication tag with AuthSend, the ARC scheme calls RC's Send on a dummy message to generate a ciphertext ct that acts as authentication tag. AuthReceive is implemented as a Receive call on this authentication tag/ciphertext.

RC-ARC.Setup(1^λ)

1 : **return** RRC.Setup(1^λ)

RC-ARC.Init(pp)

1 : **return** RRC.Init(pp)

RC-ARC.Send($\mathsf{st}_\mathcal{P}$, ad, pt)

1 : $\mathsf{ct}' \leftarrow$ RRC.Send($\mathsf{st}_\mathcal{P}$, ad, pt)
2 : $\mathsf{ct} \leftarrow (0, \mathsf{ct}')$
3 : **return** ct

RC-ARC.Receive($\mathsf{st}_\mathcal{P}$, ad, ct)

1 : $(b, \mathsf{ct}') \leftarrow \mathsf{ct}$
2 : **if** $b \neq 0$ **then return** $(\mathsf{false}, \perp, \perp, \perp)$
3 : **return** RRC.Receive($\mathsf{st}_\mathcal{P}$, ad, ct')

RC-ARC.AuthSend($\mathsf{st}_\mathcal{P}$)

1 : $(\mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{ct}) \leftarrow$ RRC.Send($\mathsf{st}_\mathcal{P}, 0, 0$)
2 : **return** $(\mathsf{st}'_\mathcal{P}, \mathsf{num}, (1, \mathsf{ct}))$

RC-ARC.AuthReceive($\mathsf{st}_\mathcal{P}$, at)

1 : $(b, \mathsf{at}') \leftarrow \mathsf{at}$
2 : **if** $b \neq 1$ **then return** $(\mathsf{false}, \perp, \perp)$
3 : $(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow$ RRC.Receive($\mathsf{st}_\mathcal{P}, 0, \mathsf{at}'$)
4 : **return** $(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num})$

Figure 2.9: UNF-secure ARC scheme based on a RID-secure RC scheme RRC.

This approach enables Theorem 12, which also implies that the scheme of Figure 2.9 is also r-UNF- and s-UNF-secure, since the theorem states that the scheme of Figure 2.9 is UNF-secure.

Theorem 12. Let RRC be a RC scheme and RC-ARC be the ARC scheme built on RRC as Figure 2.9 shows. If RRC is RID-secure, ORDINALS-secure and correct, then RC-ARC is UNF-, ORDINALS-secure and correct.

Proof. Correctness follows from the correctness of the underlying scheme RRC and the use of domain separation for tags and ciphertexts. ORDINALS-security follows from the construction.

Now, we sketch the proof that RID security of RRC implies UNF security of RC-ARC. For any adversary $\mathcal{A}$ playing the UNF game with RC-ARC, we build a RID adversary $\mathcal{B}$ for RRC. Each query made by $\mathcal{A}$ to the oracles SEND, RECEIVE, EXP_pt, EXP_st are forwarded by $\mathcal{B}$ to its own corresponding oracles (and domain separation is correctly implemented where needed). Queries of the form AUTHSEND($\mathcal{P}$) are simulated by $\mathcal{B}$ querying $\mathsf{at}' \leftarrow \mathsf{SEND}(\mathcal{P}, 0, 0)$ and setting $\mathsf{at} \leftarrow (1, \mathsf{at}')$, which perfectly simulates the generation of a tag in ARC. Finally, AUTHRECEIVE queries are simulated using the RECEIVE oracle on the tag/ciphertext. $\mathcal{B}$ can perfectly simulate the UNF game for $\mathcal{A}$.

Now, let us assume that the UNF adversary $\mathcal{A}$ wins with the forgery and bad-P predicates evaluating to true. It means a forgery was received by a party $\mathcal{P}$, then, later, that party sent a tag (i.e. a ciphertext in the RID game played by $\mathcal{B}$) that is honestly and successfully delivered to a party $\overline{\mathcal{P}}$. That implies that in the RID game played by $\mathcal{B}$, a party received a

forgery, then sent a message that was successfully delivered, which is a winning condition for $\mathcal{B}$.

The second case is when the UNF adversary $\mathcal{A}$ wins with the forgery and bad-$\overline{\mathsf{P}}$ predicates evaluating to true. This means that a forgery was received by a party $\mathcal{P}$ with ordinal num, then a tag with ordinal $\mathsf{num}' \geq \mathsf{num}$ was successfully received. Note that in our RC-ARC construction the tags are ciphertexts, thus the ordinals are strictly increasing, i.e., $\mathsf{num}' > \mathsf{num}$. Therefore, in the RID game played by $\mathcal{B}$, a forgery with ordinal num was received by $\mathcal{P}$, then later a honest ciphertext with ordinal $\mathsf{num}' > \mathsf{num}$ was successfully delivered to $\mathcal{P}$, making the bad-$\overline{\mathsf{P}}$ predicate in the RID game true.

Hence, for any adversary $\mathcal{A}$ winning the UNF game, there exists a RID adversary $\mathcal{B}$ that wins with at least the same probability. □

2.4.2 UNF-secure ARC from any RC

We present a non-optimized UNF-secure ARC scheme given a RC scheme (Definition 1), i.e. we define the two additional algorithms AuthSend and AuthReceive. We present our scheme in Figure 2.10.

Scheme description

The Setup procedure runs the corresponding procedure of the underlying RC scheme. Init initializes the states for parties A and B. The Send and Receive procedures call the respective procedures of the underlying RC scheme. The Send procedure stores the hash of $(\mathsf{ad}, \mathsf{ct})$ for the message being sent, together with the corresponding num that the underlying RC.Send algorithm returns. The tuple composed of num and the hash is stored in a set S, which is in turn stored in the internal state of the party. The Send algorithm also updates the ordinal num in the state. The Receive procedure verifies if the RC.Receive algorithm accepts the inputs and that the received message is not a forgery on a previously authenticated message, which is by construction contained in $\mathsf{S}_{\mathsf{ack}}$. If both checks pass, Receive stores the hash of $(\mathsf{ad}, \mathsf{ct})$ together with the ordinal num returned by RC.Receive in a set R.

AuthSend puts in the authentication tag at the hashes of the sent and received messages along with the last num returned by RC.Send. Since the adversary can reorder messages in both the normal and out-of-band channels, the ordinal num indicates to the recipient of the tag which messages to compare against at. AuthReceive parses the authentication tag and checks whether the messages received by the counterpart are in the local S set. Then it verifies whether the local set of received messages, without the messages not encompassed by at, is a subset of the messages sent by the counterpart. If one of these

ARC.Setup(1^λ)

1 : $\mathsf{pp}_0 \leftarrow \mathsf{RC.Setup}(1^\lambda)$
2 : $\mathsf{hk} \leftarrow \mathsf{H.KGen}(1^\lambda)$
3 : **return** $(\mathsf{pp}_0, \mathsf{hk})$

ARC.Init(pp)

1 : $(\mathsf{pp}_0, \mathsf{hk}) \leftarrow \mathsf{pp}$
2 : $(\mathsf{st}'_\mathsf{A}, \mathsf{st}'_\mathsf{B}, z') \leftarrow \mathsf{RC.Init}(\mathsf{pp}_0)$
3 : $\mathsf{num}, \mathsf{max\text{-}num} \leftarrow \bot;\ \mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack} \leftarrow \emptyset$
4 : $\mathsf{st}_\mathsf{A} \leftarrow (\mathsf{st}'_\mathsf{A}, \mathsf{hk}, \mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \mathsf{num}, \mathsf{max\text{-}num})$
5 : $\mathsf{st}_\mathsf{B} \leftarrow (\mathsf{st}'_\mathsf{B}, \mathsf{hk}, \mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \mathsf{num}, \mathsf{max\text{-}num})$
6 : $z \leftarrow (z', \mathsf{pp})$
7 : **return** $(\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z)$

ARC.Send($\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{pt}$)

1 : $(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \mathsf{S}, \cdot, \cdot, \cdot) \leftarrow \mathsf{st}_\mathcal{P}$
2 : $(\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{ct}) \leftarrow \mathsf{RC.Send}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}, \mathsf{pt})$
3 : $h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$
4 : $\mathsf{st}_\mathcal{P}.\mathsf{S} \leftarrow \mathsf{S} \cup \{(\mathsf{num}, h)\}$
5 : $\mathsf{st}_\mathcal{P}.\mathsf{num} \leftarrow \mathsf{num}$
6 : **return** $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{ct})$

ARC.AuthSend($\mathsf{st}_\mathcal{P}$)

1 : $(\cdot, \cdot, \mathsf{S}, \mathsf{R}, \cdot, \mathsf{num}, \cdot) \leftarrow \mathsf{st}_\mathcal{P}$
2 : $\mathsf{at} \leftarrow (\mathsf{S}, \mathsf{R}, \mathsf{num})$
3 : **return** $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{at})$

ARC.Receive($\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{ct}$)

1 : $(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \cdot, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \cdot, \mathsf{max\text{-}num}) \leftarrow \mathsf{st}_\mathcal{P}$
2 : $(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{RC.Receive}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}, \mathsf{ct})$
3 : **if** $\neg\mathsf{acc}$ **then**
4 : **return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
5 : $h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$
6 : **if** $\mathsf{num} \leq \mathsf{max\text{-}num} \wedge (\mathsf{num}, h) \notin \mathsf{S}_\mathsf{ack}$ **then**
7 : **return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
8 : $\mathsf{st}_\mathcal{P}.\mathsf{R} \leftarrow \mathsf{R} \cup \{(\mathsf{num}, h)\}$
9 : $\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
10 : **return** $(\mathsf{acc}, \mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{pt})$

ARC.AuthReceive($\mathsf{st}_\mathcal{P}, \mathsf{at}$)

1 : $(\cdot, \cdot, \mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \mathsf{num}, \mathsf{max\text{-}num}) \leftarrow \mathsf{st}_\mathcal{P}$
2 : $(\mathsf{S}^{\overline{\mathcal{P}}}, \mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{num}^{\overline{\mathcal{P}}}) \leftarrow \mathsf{at}$
3 : // $\overline{\mathcal{P}}$ received a forgery
4 : **if** $\mathsf{R}^{\overline{\mathcal{P}}} \not\subseteq \mathsf{S}$ **then**
5 : **return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \mathsf{num})$
6 : $\mathsf{R}^\mathcal{P}_\subseteq \leftarrow \{(\mathsf{num}, \cdot) \in \mathsf{R} : \mathsf{num} \leq \mathsf{num}^{\overline{\mathcal{P}}}\}$
7 : // $\mathcal{P}$ received a forgery
8 : **if** $\mathsf{R}^\mathcal{P}_\subseteq \not\subseteq \mathsf{S}^{\overline{\mathcal{P}}}$ **then return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \mathsf{num})$
9 : $\mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{ack} \leftarrow \mathsf{S}_\mathsf{ack} \cup \mathsf{S}^{\overline{\mathcal{P}}}$
10 : $\mathsf{st}_\mathcal{P}.\mathsf{max\text{-}num} \leftarrow \max\{\mathsf{max\text{-}num}, \mathsf{num}^{\overline{\mathcal{P}}}\}$
11 : **return** $(\mathsf{true}, \mathsf{st}_\mathcal{P}, \mathsf{num}^{\overline{\mathcal{P}}})$

Figure 2.10: UNF-secure ARC scheme based on a RC scheme RC (Definition 1). The scheme uses four additional variables compared to RC: S is the set of sent (num, h); R is the set of received (num, h); S_ack is the set of (num, h) which are expected to be received (according to the received authentication tag at); max-num represents the largest num received in an at. All sets are append-only. For simplicity of exposition, we omit the optimisation where R is sent as a single hash and n ordinals as done in Figure 2.7 for RID security.

conditions is not satisfied, then a forgery is detected. The sent messages authenticated by the counterpart are stored in a set $\mathsf{S}_{\mathsf{ack}}$. The procedure Receive uses this set to avoid forgeries on already authenticated num's.

The size of the authentication tags and the state of each party in the scheme of Figure 2.10 is linear in the number of sent and received messages. Messages can nevertheless be efficiently exchanged out-of-band in practice, e.g., using Bluetooth. Otherwise, parties can send authentication information over the insecure channel and authenticate it using the out-of-band channel by hashing and comparing digests [PV06]. If we assume that the underlying network channel is ordered (e.g., by using TCP), then the hashes of the last sent and received messages suffice to detect forgeries [CDV21].

Security analysis

We now analyze the security properties of the scheme in Figure 2.10. Correctness of the scheme follows from the correctness of the underlying RC scheme. Similarly, ORDINALS security follows from the ORDINALS security of RC, as the scheme of Figure 2.10 outputs the same num that RC outputs.

The UNF-security of the ARC scheme presented in Figure 2.10, lies in the collision resistance of the hash function that the scheme uses. When one party $\mathcal{P}$ wants to authenticate the communication it produces an authentication tag containing the hashes of all the messages inboxed and outboxed by $\mathcal{P}$. These hashes can be compared with the counterpart $\overline{\mathcal{P}}$ to detect if any forgery has been received and accepted by one of the participants. In what follows we prove that the scheme of Figure 2.10 is UNF-secure.

Theorem 13 (Unforgeability of ARC)**.** Let H be a (t_{cr}, ϵ_{cr})-collision resistant hash function (Definition 6). Then the ARC scheme, that we present in Figure 2.10, is (q, t, ϵ_{cr})-UNF secure ARC scheme where $t \approx t_{cr}$.

Proof. Assume an adversary $\mathcal{A}$ playing the UNF game (Figure 2.8), which makes at most q oracle queries and runs in time at most t. We assume the advantage of $\mathcal{A}$ is ϵ, hence by Definition 11 we have $\Pr[\mathsf{UNF}^{\mathcal{A}}(1^{\lambda}) \Rightarrow 1] = \epsilon$. We construct an adversary $\mathcal{B}$, running in time approximately equal to t, which, running $\mathcal{A}$ as a subroutine, wins the collision resistance game for H (Definition 7), that is

$$\Pr[\mathsf{CR}^{\mathcal{B}^{\mathcal{A}}}(1^{\lambda}) \Rightarrow 1] = 1.$$

The UNF adversary $\mathcal{A}$ wins when one party accepts a forgery (predicate forgery) and at least one of the two parties fails to detect the forgery (predicates bad-P and bad-$\overline{\mathsf{P}}$). Suppose there exist $\mathcal{P}$, num, num$'$, ad, ct, at, x, y such that $\mathsf{forgery}(\log, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x) = \mathsf{true}$. We analyze the predicates bad-P and bad-$\overline{\mathsf{P}}$ separately, starting with the latter.

CR adversary $\mathcal{B}^{\mathcal{A}}(\mathsf{hk})$

$(1^\lambda, \mathsf{hk}_0) \leftarrow \mathsf{hk}$; $\mathsf{pp} \leftarrow \mathsf{ARC.Setup}(1^\lambda)$; $(\mathsf{pp}_0, \mathsf{hk}') \leftarrow \mathsf{pp}$; $\mathsf{pp}' \leftarrow (\mathsf{pp}_0, \mathsf{hk})$
$(\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z) \leftarrow \mathsf{ARC.Init}(\mathsf{pp}')$
$\mathsf{state}[\cdot], \mathsf{plaintext}[\cdot], \log[\cdot], \mathsf{auth}[\cdot], \mathsf{st}_* \leftarrow \bot$; $i \leftarrow 0$
$\mathcal{A}^{\mathcal{O}}(z)$
if $\exists\ \mathsf{num}, \mathcal{P}, \mathsf{ad}, \mathsf{ct}, \mathsf{ad}', \mathsf{ct}'$:
  $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \in \log \wedge (\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}', \mathsf{ct}') \wedge (\mathsf{ad}, \mathsf{ct}) \neq (\mathsf{ad}', \mathsf{ct}')$ **then**
    return $(\mathsf{ad}, \mathsf{ct}), (\mathsf{ad}', \mathsf{ct}')$
else abort

Figure 2.11: Adversary $\mathcal{B}$ where $\mathcal{O} = \{\mathsf{SEND}, \mathsf{RECEIVE}, \mathsf{AUTHSEND}, \mathsf{AUTHRECEIVE}, \mathsf{EXP}_\mathsf{pt}, \mathsf{EXP}_\mathsf{st}\}$ for the proof of Theorem 13.

The forgery predicate states that $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}', \mathsf{ct}') \in \log$ and $(\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \log[x]$ for some $x \in \mathbb{N}$, where $(\mathsf{ad}', \mathsf{ct}') \neq (\mathsf{ad}, \mathsf{ct})$. This means that $(\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}', \mathsf{ct}'))) \in \mathsf{S}^{\mathcal{P}}$ and $(\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))) \in \mathsf{R}^{\overline{\mathcal{P}}}$, otherwise the forgery is trivially detected because $(\mathsf{num}, \cdot) \notin \mathsf{R}^{\overline{\mathcal{P}}}$. Moreover, by the bad-$\overline{\mathsf{P}}$ predicate we know that $\mathsf{R}^{\overline{\mathcal{P}}}_{\subseteq} \subseteq \mathsf{S}^{\mathcal{P}}$ for any $\mathsf{num} \leq \mathsf{num}'$, which implies that $(\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}', \mathsf{ct}'))) = (\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct})))$. By correctness, $(\mathsf{num}, \cdot)$ can appear only once in $\mathsf{S}^{\mathcal{P}}$, respectively in $\mathsf{R}^{\overline{\mathcal{P}}}$, and by assumption $(\mathsf{ad}', \mathsf{ct}') \neq (\mathsf{ad}, \mathsf{ct})$, therefore we have a collision for $\mathsf{H.Eval}(\mathsf{hk}, \cdot)$.

We now analyze the bad-P predicate. The forgery predicate states that $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}', \mathsf{ct}') \in \log$ and $(\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \log[x]$ for some $x \in \mathbb{N}$, where $(\mathsf{ad}', \mathsf{ct}') \neq (\mathsf{ad}, \mathsf{ct})$, otherwise the forgery is trivially detected. This implies that $(\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct})) \in \mathsf{R}^{\overline{\mathcal{P}}}$ when $\mathsf{ARC.Receive}(\cdot, \mathsf{ad}, \mathsf{ct}) \rightarrow (\mathsf{true}, \cdot, \mathsf{num}, \cdot)$. By the bad-P predicate we know that $\overline{\mathcal{P}}$ sends an authentication tag at after accepting $(\mathsf{ad}, \mathsf{ct})$, since $(\text{“authsend”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{at}) = \log[y]$ and $y > x$, which means that $(\mathsf{num}, \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$ is in the $\mathsf{R}^{\overline{\mathcal{P}}}$ that at contains. The rest of the argument follows the same approach as the previous paragraph.

Figure 2.11 describes the adversary $\mathcal{B}$ which plays against the collision resistance game. $\mathcal{B}$ runs the ARC.Setup procedure and replaces the hash key hk' that the procedure returns with the hk that the adversary receives from the CR challenger. After running $\mathcal{A}$ as a subroutine, $\mathcal{B}$ analyzes the log array to find the $(\mathsf{ad}, \mathsf{ct})$, $(\mathsf{ad}', \mathsf{ct}')$ pairs that represents a forgery and returns those. If $\mathcal{A}$ wins the UNF game, then $\mathcal{B}$ wins the CR, that is

$$\Pr[\mathsf{CR}^{\mathcal{B}}(1^\lambda) \Rightarrow 1] \geq \Pr[\mathsf{UNF}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] = \epsilon.$$

Moreover, $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and executes an additional negligible amount of work, so $t \approx t_{cr}$. □

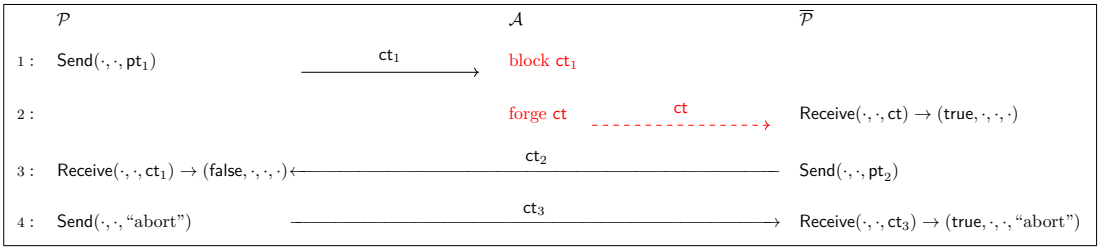

Figure 2.12: Visualising r-RID security after one honest round trip. For clarity, associated data and certain inputs and outputs of the Send and Receive algorithms are omitted. We denote the adversary by $\mathcal{A}$.

2.5 Performance and security trade-offs

The extended abstract corresponding to this chapter demonstrates [BCC+23a, Section 6] that r-RID and r-UNF impose linear communication complexity for RC and ARC schemes (i.e., the ciphertext and tag lengths grow linearly in the number of sent messages and security parameter) in the worst-case scenario, where one party continuously sends messages without receiving any from their counterpart. This section explores methods to bypass these lower bounds and propose practical protocols for active attack detection.

We first recall (Section 2.5.1) that s-RID and s-UNF provide delayed r-RID/r-UNF guarantees, i.e., protocols achieve r-RID and r-UNF security after one honest round trip. We then argue (Section 2.5.2) that protocols can achieve s-RID/s-UNF security at a significantly lower cost than their r-RID/r-UNF counterparts. As ciphertext sizes remain unbounded in the schemes presented so far, we propose two pruning approaches to reduce communication overhead. The first, presented in Section 2.5.3, leverages communication epochs [ACD19] to prune after each full round trip the set of received messages that that parties send to achieve s-RID security. In Section 2.5.4, we introduce an optimised scheme for UNF security that prunes redundant messages from authentication tags, taking advantage of the authenticity of the out-of-band channel. We conclude the section with a lightweight three-move protocol that builds on s-UNF's delayed r-UNF-like guarantees to authenticate communication bidirectionally over the out-of-band authenticated channel (Section 2.5.5).

2.5.1 Delayed r-RID/r-UNF security from s-RID/s-UNF security

In this section we explain how s-RID provides r-RID after a honest round trip. While focus here on security notions for the in-band channel, similar arguments apply to the out-of-band channel.

Figure 2.12 illustrates the delayed r-RID guarantees that s-RID offers. Suppose parties use a s-RID-secure scheme, which ensures that $\mathcal{P}$ can detect that $\overline{\mathcal{P}}$ received a forged

message. Assume that $\overline{\mathcal{P}}$ receives and accepts a forged message (line 2 in Figure 2.12) and subsequently sends a message to $\mathcal{P}$ (line 3). Given s-RID security, $\mathcal{P}$ detects that $\overline{\mathcal{P}}$ received a forgery (thus, $\mathcal{P}$'s call to Receive on line 3 rejects). If upon detecting that $\overline{\mathcal{P}}$ was victim of a forgery, $\mathcal{P}$ sends an "abort" message to $\overline{\mathcal{P}}$ (line 4), $\overline{\mathcal{P}}$ is able to decrypt the message, detect the forgery and abort the communication after one honest round-trip of messages.

This informal argument indicates that s-RID-secure RC schemes can be transformed (by adding an "abort" message) into RC schemes that ensure a weaker form of r-RID security: r-RID after a honest round-trip. Additional care is needed to formalize this property and implement it in practice; for instance, if $\mathcal{P}$ rejects ct_2 in Figure 2.12 due to a network-induced bit-flip, the application should avoid a false-positive communication abort. This is approach is noteworthy since s-RID imposes less overhead compared to r-RID, a point we will elaborate in the next section. Furthermore, the tree-move out-of-band bidirectional authentication protocol that we present in Section 2.5.5 builds on this concept of delayed r-UNF-like security, which we formalize to design a practical scheme.

2.5.2 Practicality of s-RID and s-UNF security

Security notion	r-RID	s-RID
Overhead	$\mathcal{O}(n\lambda + c_n)$	$\mathcal{O}(\lambda + c_n)$
Optimized overhead (Section 2.5.3)	N/A	$\mathcal{O}(\lambda + c_{n_{\mathsf{fresh}}})$

Table 2.1: Overhead induced by the two RID security notions. We assume that n messages are received and c_i is the space needed to encode i ordinals. We indicate the encoding space with a different variable since compression mechanisms such as run-length encoding can be used in this case. The variable n_{fresh} refers to the number of messages received in the last two epochs.

We focus here on s-RID security, but similar arguments hold for s-UNF security. The RRC scheme in Figure 2.7 achieves s-RID security by sending to the counter part the list of received nums and an *hash* of the R set. Informally, this suffices for security because a party can immediately detect when their counterpart has received a forgery (by keeping in state their sent messages and recomputing the hash). Table 2.1 summarizes the overheads of the two notions together with the optimization presented in Section 2.5.3, where we show that it is enough to only send information about messages received during the two last epochs. This significantly reduces the overhead for the scenarios in which the communication is "balanced".

One can reduce ciphertext size further by optimising for the "good case" scenario where messages are delivered in-order; in this case, ordinals can be encoded in ranges. For epochs with no lost messages, it suffices to encode only the last index. In any case, as the size of

a single message in today's secure messaging applications can be several kilobytes, the overhead that s-RID imposes seems reasonable. We leave a deeper and concrete analysis to determine the impact of RID/UNF security in practice to future work.

In Figure 2.7, the entire set of received messages is hashed (using a regular hash function) every time a message is sent by $\mathcal{P}$. Consequently, when $\mathcal{P}$ receives a new message, the entire hash must be re-computed when $\mathcal{P}$ sends their next message. To avoid this, the scheme can use an *incremental hash function* [BGG94, CDvD$^+$03] such that, when a message is received, an efficient operation only depending on the new message and the previous digest can be executed to derive the new digest. Hash digests can be as small as a group element [CDvD$^+$03]. This enables parties to prune their set of sent/received messages in state. For example, if $\mathcal{P}$ receives a message m claiming that $\overline{\mathcal{P}}$ has received the first k messages from $\mathcal{P}$, and $\mathcal{P}$ has received messages for all possible ordinals that precede the ordinal of m, then $\mathcal{P}$ can safely store just the incrementally-hashed value corresponding to the first k messages, since $\overline{\mathcal{P}}$ can no longer claim to have only received a strict subset of the k messages.

Remark 14. The RID-secure RC of Figure 2.7 sends the set of *received* ordinals for authentication (line 2 of the RRC.Send algorithm). Since ordinals are elements of a set on which a total order is defined, a simple optimization—that could reduce the overhead by up to 50%—consists in sending the smallest set among the set of received ordinals *or* the set of not received ordinals alongside a bit indicating which type of set has been sent. This optimization applies to all schemes that send sets of ordinals.

2.5.3 Epoch-based optimisation for s-RID security

In this section, we describe how to design an optimized s-RID-secure RC scheme based on a correct and ORDINALS-secure RC scheme. The formal description of the optimized s-RID-secure RC scheme is given in Figure 2.13. We begin with a high-level description of the optimization.

The parties start at $\mathsf{epoch} = 0$ and $\mathsf{epoch} = 1$, respectively. Each time a party sends a message, they attach their current epoch to it (line 4 of the Send procedure in Figure 2.13) Upon receiving a message, a party $\mathcal{P}$ with $\mathsf{epoch} = t$ checks whether the attached epoch from the other party $\overline{\mathcal{P}}$ is at most $t + 1$, that is $\mathsf{epoch}_{\overline{\mathcal{P}}} \leq t + 1$, accounting for possible out-of-order messages from earlier epochs (see the checks function in Figure 2.13). If the received epoch equals $t + 1$, i.e, $\overline{\mathcal{P}}_{\mathsf{epoch}} = t + 1$, the party $\mathcal{P}$ updates their epoch to $t + 2$ (lines 12 and 13 and the Receive function). This "ping-pong" style of updating epochs mirrors the Double Ratchet's asymmetric ratchet [PM16], ensuring the parties' epoch values remain one apart as they communicate.

To achieve s-RID security, whenever sending a message with $\mathsf{epoch} = t$, the sender attaches the accumulated hash and the corresponding nums of all messages received during the time

their epoch was t and $t-2$. This means that the parties report only the messages from the current and last epoch, as opposed to all previous messages, improving efficiency compared to the original s-RID construction. Intuitively, the "ping-pong" updating of epochs enables the parties to regularly inform each other about the messages they receive, allowing them to reliably consider those messages authenticated. In other words, a complete round-trip of honest messages implicitly acknowledges the reception and verification of the hash and associated nums. In Section 2.5.4 we use a similar mechanism with explicit acknowledgments in the out-of-band channel to design our three-move bidirectional authentication protocol.

We now state the main theorem of this section, with a proof sketch provided in Appendix A.2. The proof strategy relies on the fact that epoch values only increase when both parties have received a message. An adversary cannot force a party to skip epochs, as a receiver $\mathcal{P}$ only accepts messages with $\mathsf{epoch}_{\overline{\mathcal{P}}} \leq t+1$. This ensures that conveying information about the messages received in the last two epochs is sufficient to achieve s-RID security.

Theorem 15. Let H be a (t_{cr}, ϵ_{cr})-collision resistant hash function. Then s-RID-RC (Figure 2.13) is a (q, t, ϵ_{cr})-s-RID-secure RC where $t_{cr} \approx t$ and q is upper bounded by t.

2.5.4 Reducing bandwidth for UNF security

In Figure 2.14, we present a scheme that optimizes bandwidth consumption for UNF security by leveraging the authenticity of the out-of-band channel, which prevents the adversary from forging messages in the out-of-band channel. Suppose that $\mathcal{P}$ sends an authentication tag to $\overline{\mathcal{P}}$, then $\overline{\mathcal{P}}$ acknowledges reception of the tag to $\mathcal{P}$. At this point, $\mathcal{P}$ no longer needs to send the information that $\overline{\mathcal{P}}$ has already obtained. Our scheme supports out-of-order communication even on the authenticated channels. However, this approach requires parties to keep track of, e.g., which tags their partner has received to determine what is safe to prune from state (in S_{at}-Seen).

Scheme description

The Send procedure stores the hash of $(\mathsf{ad}, \mathsf{ct})$ for the message being sent, together with the corresponding num that the underlying RC.Send algorithm returns. The algorithm stores (num, h) in a set S, which the schemes stores in the party's internal state. The Send algorithm also updates the ordinal num in the state.

The Receive procedure verifies whether the RC.Receive algorithm accepts the inputs and verifies that the received message is not a forgery on a previously authenticated message, which is by construction contained in $\mathsf{S}_{\mathsf{ack}}$. If both checks pass, Receive stores the hash of $(\mathsf{ad}, \mathsf{ct})$ together with the ordinal num returned by RC.Receive in a set R.

s-RID-RC.Setup(1^λ)

$\mathsf{pp}_0 \leftarrow \mathsf{RC.Setup}(1^\lambda)$
$\mathsf{hk} \leftarrow \mathsf{H.KGen}(1^\lambda);\ \mathsf{hk}' \leftarrow \mathsf{H.KGen}(1^\lambda)$
$\mathsf{pp} \leftarrow (\mathsf{pp}_0, \mathsf{hk}, \mathsf{hk}')$
return pp

s-RID-RC.Init(pp)

$(\mathsf{pp}_0, \mathsf{hk}, \mathsf{hk}') \leftarrow \mathsf{pp}$
$(\mathsf{st}'_\mathsf{A}, \mathsf{st}'_\mathsf{B}, z') \leftarrow \mathsf{RC.Init}(\mathsf{pp}_0)$
$\mathsf{epoch} \leftarrow 0$
$\mathsf{S}, \mathsf{R}_{\mathsf{curr}}, \mathsf{R}_{\mathsf{prev}} \leftarrow \emptyset$
$\mathsf{st}_\mathsf{A} \leftarrow (\mathsf{st}'_\mathsf{A}, \mathsf{hk}, \mathsf{hk}', \mathsf{S}, \mathsf{R}_{\mathsf{curr}}, \mathsf{R}_{\mathsf{prev}}, \mathsf{epoch})$
$\mathsf{st}_\mathsf{B} \leftarrow (\mathsf{st}'_\mathsf{B}, \mathsf{hk}, \mathsf{hk}', \mathsf{S}, \mathsf{R}_{\mathsf{curr}}, \mathsf{R}_{\mathsf{prev}}, \mathsf{epoch}+1)$
$z \leftarrow (z', \mathsf{pp})$
return $(\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z)$

s-RID-RC.Send($\mathsf{st}_\mathcal{P}$, ad, pt)

$(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \mathsf{hk}', \mathsf{S}, \mathsf{R}_{\mathsf{curr}}, \mathsf{R}_{\mathsf{prev}}, \mathsf{epoch}) \leftarrow \mathsf{st}_\mathcal{P}$
$\mathsf{nums}' \leftarrow \{\mathsf{num}' : (\mathsf{num}', \_) \in \mathsf{R}_{\mathsf{curr}} \cup \mathsf{R}_{\mathsf{prev}}\}$
$\mathsf{R}' \leftarrow (\mathsf{nums}', \mathsf{H.Eval}(\mathsf{hk}', \mathsf{R}_{\mathsf{curr}} \cup \mathsf{R}_{\mathsf{prev}}))$
$\mathsf{ad}' \leftarrow (\mathsf{ad}, \mathsf{epoch}, \mathsf{R}')$
$(\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{ct}') \leftarrow \mathsf{RC.Send}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}', \mathsf{pt})$
$\mathsf{ct} \leftarrow (\mathsf{ct}', \mathsf{epoch}, \mathsf{R}')$
$h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$
$\mathsf{st}_\mathcal{P}.\mathsf{S} \leftarrow \mathsf{S} \cup \{(\mathsf{num}, h)\}$
return $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{ct})$

s-RID-RC.Receive($\mathsf{st}_\mathcal{P}$, ad, ct)

$(\mathsf{ct}', \mathsf{epoch}^{\overline{\mathcal{P}}}, \mathsf{R}^{\overline{\mathcal{P}}}) \leftarrow \mathsf{ct}$
$(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \mathsf{hk}', \mathsf{S}, \mathsf{R}_{\mathsf{curr}}, \mathsf{R}_{\mathsf{prev}}, \mathsf{epoch}) \leftarrow \mathsf{st}_\mathcal{P}$
$\mathsf{ad}' \leftarrow (\mathsf{ad}, \mathsf{epoch}^{\overline{\mathcal{P}}}, \mathsf{R}^{\overline{\mathcal{P}}})$
$(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{RC.Receive}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}', \mathsf{ct}')$
if $\neg\mathsf{acc}$ **then return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
$h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$
if $\mathsf{checks}(\mathsf{st}_\mathcal{P}, \mathsf{ct}, h, \mathsf{num})$ **then**
  return $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
$\mathsf{st}_\mathcal{P}.\mathsf{R}_{\mathsf{curr}} \leftarrow \mathsf{R}_{\mathsf{curr}} \cup \{(\mathsf{num}, h)\}$
$\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
// Advance epochs accordingly
if $\mathsf{epoch}^{\overline{\mathcal{P}}} = \mathsf{st}_\mathcal{P}.\mathsf{epoch} + 1$ **then**
  $\mathsf{st}_\mathcal{P}.\mathsf{epoch} \leftarrow \mathsf{st}_\mathcal{P}.\mathsf{epoch} + 2$
  $\mathsf{st}_\mathcal{P}.\mathsf{R}_{\mathsf{prev}} \leftarrow \mathsf{R}_{\mathsf{curr}}$
  $\mathsf{st}_\mathcal{P}.\mathsf{R}_{\mathsf{curr}} \leftarrow \emptyset$
return $(\mathsf{acc}, \mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{pt})$

checks($\mathsf{st}_\mathcal{P}$, ct, h, num)

$(\mathsf{nums}', h') \leftarrow \mathsf{ct.R}$
$\mathsf{epoch}' \leftarrow \mathsf{ct.epoch}$
if $\mathsf{epoch}' > \mathsf{st}_\mathcal{P}.\mathsf{epoch} + 1$ **then**
  s-bool $\leftarrow$ **true**
$\mathsf{R}^* \leftarrow \{(\mathsf{num}', \_) \in \mathsf{st}_\mathcal{P}.\mathsf{S} : \mathsf{num}' \in \mathsf{nums}')\}$
s-bool $\leftarrow$ s-bool $\vee$ $(\mathsf{H.Eval}(\mathsf{st}_\mathcal{P}.\mathsf{hk}', \mathsf{R}^*) \neq h')$
return s-bool

Figure 2.13: Optimized s-RID-secure RC scheme given a correct and ORDINALS-secure RC scheme. The set $\mathsf{R}_{\mathsf{prev}}$ is the set of (num, h) received in the previous epoch, while $\mathsf{R}_{\mathsf{curr}}$ is the set of (num, h) that the party receives in the current epoch. The variable epoch indicates the current epoch for each party.

The AuthSend procedure is similar to the unoptimized one, except that (1) it stores the set of sent messages S authenticated *within the current* at into an array S_{at}, indexed by counter $\mathsf{cnt}_{\mathsf{at}}$, and (2) it empties the set S_{at}-Seen, which is already in at and whose goal is to communicate to the other parties which authentication tags the AuthReceive function processed, as explained later.

AuthReceive behaves like ARC.AuthReceive, but with optimizations. It firstly verifies whether ${\mathsf{cnt}_{\mathsf{at}}}^{\overline{\mathcal{P}}} \leq \mathsf{max\text{-}cnt}_{\mathsf{at}}$. The goal of this check is to avoid processing old authentication tags, since AuthReceive already authenticated their content with the newer (in terms of ${\mathsf{cnt}_{\mathsf{at}}}^{\overline{\mathcal{P}}}$) tag. The max-num-at keeps indeed track of the most recent authentication tag that the procedure processed. In other words, older tags either contain less information than newer, already accepted tags or they contain outdated information that has already been verified and pruned. AuthReceive also performs garbage collection. It first stores the counter of the input at into S_{at}-Seen, which will be sent to the counterparty in the next call to AuthSend. Then it removes the already authenticated messages from memory. The party already authenticated the subset $\mathsf{R}_{\subseteq}^{\mathcal{P}}$ and it can remove the corresponding messages from the set R, which represents now the set of currently unauthenticated received messages. Similarly, AuthReceive uses the set of authentication tags that the counterpart already processed to prune the set of sent messages. In detail, the pruning of sent messages works as follows.

When a party $\mathcal{P}$ receives a set of messages ${\mathsf{S}_{\mathsf{at}}}^{\overline{\mathcal{P}}}[\mathsf{cnt}_{\mathsf{at}}] \leftarrow \mathsf{S}$ sent by the counterpart with the authentication tag number $\mathsf{cnt}_{\mathsf{at}}$, it stores them in a set $\mathsf{S}_{\mathsf{ack}}$. Then, when $\mathcal{P}$ sends a subsequent authentication tag back to $\overline{\mathcal{P}}$, it informs $\overline{\mathcal{P}}$ that the authentication tag $\mathsf{cnt}_{\mathsf{at}}$ was received using the $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\mathcal{P}}$ set. When this tag is delivered, $\overline{\mathcal{P}}$ can remove the acknowledged messages ${\mathsf{S}_{\mathsf{at}}}^{\overline{\mathcal{P}}}[\mathsf{cnt}_{\mathsf{at}}]$ for all counters in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\mathcal{P}}$ from its set $\mathsf{S}^{\overline{\mathcal{P}}}$. This reduces the size of the authentication tag as, on every round-trip on the out-of-band channel, all authenticated messages can be removed from the sets S and R. To reduce the size even further, we can use hashing optimization for the received set. Instead of sending $\mathsf{R} = \{(\mathsf{num}_1, h_1), \ldots, (\mathsf{num}_k, h_k)\}$ in AuthSend, one can send $\mathsf{R}' = \{(\mathsf{num}_1, \ldots, \mathsf{num}_k), \mathsf{H.Eval}(\mathsf{hk}, h_1, \ldots, h_k)\}$. On reception, AuthReceive can recompute the hashes of the single messages and authenticate R'. Pruning does not affect this optimization, since AuthReceive removes from S only messages that the counterpart already authenticated. A similar technique was employed by Dowling et al. [DGP22].

Security analysis

We informally argue that the scheme prunes only messages that have already been authenticated. The procedures rely on sets $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ and $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ to detect active attacks. AuthReceive prunes $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ by removing elements in $\mathsf{R}_{\subseteq}^{\mathcal{P}}$; since the procedure authenticates the elements in $\mathsf{R}_{\subseteq}^{\mathcal{P}}$ at line 11, it is safe to prune $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$. Similarly, AuthReceive prunes $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$

ARC-OP.Setup(1^λ)

$\mathsf{pp}_0 \leftarrow \mathsf{RC.Setup}(1^\lambda)$; $\mathsf{hk} \leftarrow \mathsf{H.KGen}(1^\lambda)$
return $(\mathsf{pp}_0, \mathsf{hk})$

ARC-OP.Init(pp)

$(\mathsf{pp}_0, \mathsf{hk}) \leftarrow \mathsf{pp}$
$(\mathsf{st}'_\mathsf{A}, \mathsf{st}'_\mathsf{B}, z') \leftarrow \mathsf{RC.Init}(\mathsf{pp})$
$\mathsf{num}, \mathsf{max\text{-}num}, \mathsf{cnt}_\mathsf{at}, \mathsf{max\text{-}cnt}_\mathsf{at} \leftarrow 0$
$\mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \mathsf{S}_\mathsf{at}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen} \leftarrow \emptyset$
$\mathsf{st}_\mathsf{A} \leftarrow (\mathsf{st}'_\mathsf{A}, \mathsf{hk}, \mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \mathsf{num}, \mathsf{max\text{-}num},$
  $\mathsf{cnt}_\mathsf{at}, \mathsf{max\text{-}cnt}_\mathsf{at}, \mathsf{S}_\mathsf{at}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen})$
$\mathsf{st}_\mathsf{B} \leftarrow (\mathsf{st}'_\mathsf{B}, \mathsf{hk}, \mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \mathsf{num}, \mathsf{max\text{-}num},$
  $\mathsf{cnt}_\mathsf{at}, \mathsf{max\text{-}cnt}_\mathsf{at}, \mathsf{S}_\mathsf{at}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen})$
$z \leftarrow (z', \mathsf{pp})$
return $(\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z)$

ARC-OP.Send($\mathsf{st}_\mathcal{P}$, ad, pt)

$(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \mathsf{S}, \cdots) \leftarrow \mathsf{st}_\mathcal{P}$
$(\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{ct}) \leftarrow \mathsf{RC.Send}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}, \mathsf{pt})$
$h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$
$\mathsf{st}_\mathcal{P}.\mathsf{S} \leftarrow \mathsf{S} \cup \{(\mathsf{num}, h)\}$
$\mathsf{st}_\mathcal{P}.\mathsf{num} \leftarrow \mathsf{num}$
return $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{ct})$

ARC-OP.Receive($\mathsf{st}_\mathcal{P}$, ad, ct)

$(\mathsf{st}'_\mathcal{P}, \mathsf{hk}, \cdot, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \cdot, \mathsf{max\text{-}num}, \cdots) \leftarrow \mathsf{st}_\mathcal{P}$
$(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{RC.Receive}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}, \mathsf{ct})$
if $\neg\mathsf{acc}$ **then return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
$h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$
if $\mathsf{num} \le \mathsf{max\text{-}num} \wedge (\mathsf{num}, h) \notin \mathsf{S}_\mathsf{ack}$:
  return $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
$\mathsf{st}_\mathcal{P}.\mathsf{R} \leftarrow \mathsf{R} \cup \{(\mathsf{num}, h)\}$
$\mathsf{st}_\mathcal{P}.\mathsf{st}'_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
return $(\mathsf{acc}, \mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{pt})$

ARC-OP.AuthSend($\mathsf{st}_\mathcal{P}$)

$(\cdot, \cdot, \mathsf{S}, \mathsf{R}, \cdot, \mathsf{num}, \cdot, \mathsf{cnt}_\mathsf{at}, \cdot, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen}) \leftarrow \mathsf{st}_\mathcal{P}$
$\mathsf{at} \leftarrow (\mathsf{S}, \mathsf{R}, \mathsf{num}, \mathsf{cnt}_\mathsf{at}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen})$
$\mathsf{st}_\mathcal{P}.\mathsf{cnt}_\mathsf{at} \leftarrow \mathsf{st}_\mathcal{P}.\mathsf{cnt}_\mathsf{at} + 1$
$\mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{at}[\mathsf{st}_\mathcal{P}.\mathsf{cnt}_\mathsf{at}] \leftarrow \mathsf{S}$
$\mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen} \leftarrow \emptyset$
return $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{at})$

ARC-OP.AuthReceive($\mathsf{st}_\mathcal{P}$, at)

$(\cdot, \cdot, \mathsf{S}, \mathsf{R}, \mathsf{S}_\mathsf{ack}, \mathsf{num}, \mathsf{max\text{-}num},$
  $\mathsf{cnt}_\mathsf{at}, \mathsf{max\text{-}cnt}_\mathsf{at}, \mathsf{S}_\mathsf{at}, \cdot) \leftarrow \mathsf{st}_\mathcal{P}$
$(\mathsf{S}^{\overline{\mathcal{P}}}, \mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{num}^{\overline{\mathcal{P}}}, \mathsf{cnt}_\mathsf{at}{}^{\overline{\mathcal{P}}}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}) \leftarrow \mathsf{at}$
$\mathsf{R}^\mathcal{P}_\subseteq \leftarrow \{(\mathsf{num}, \cdot) \in \mathsf{R} : \mathsf{num} \le \mathsf{num}^{\overline{\mathcal{P}}}\}$
if $\mathsf{cnt}_\mathsf{at}{}^{\overline{\mathcal{P}}} \le \mathsf{max\text{-}cnt}_\mathsf{at}$:
  $\mathsf{prune}(\mathsf{st}_\mathcal{P}, \mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{cnt}_\mathsf{at}{}^{\overline{\mathcal{P}}}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}, \mathsf{R}^\mathcal{P}_\subseteq)$
  return $(\mathsf{true}, \mathsf{st}_\mathcal{P}, \mathsf{num}^{\overline{\mathcal{P}}})$
// $\overline{\mathcal{P}}$ received a forgery
if $\mathsf{R}^{\overline{\mathcal{P}}} \not\subseteq \mathsf{S}$ **then return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \mathsf{num})$
// $\mathcal{P}$ received a forgery
if $\mathsf{R}^\mathcal{P}_\subseteq \not\subseteq \mathsf{S}^{\overline{\mathcal{P}}}$ **then return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \mathsf{num})$
$\mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{ack} \leftarrow \mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{ack} \cup \mathsf{S}^{\overline{\mathcal{P}}}$
$\mathsf{st}_\mathcal{P}.\mathsf{max\text{-}num} \leftarrow \max\{\mathsf{max\text{-}num}, \mathsf{num}^{\overline{\mathcal{P}}}\}$
$\mathsf{st}_\mathcal{P}.\mathsf{max\text{-}cnt}_\mathsf{at} \leftarrow \max\{\mathsf{cnt}_\mathsf{at}{}^{\overline{\mathcal{P}}}, \mathsf{st}_\mathcal{P}.\mathsf{max\text{-}cnt}_\mathsf{at}\}$
$\mathsf{prune}(\mathsf{st}_\mathcal{P}, \mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{cnt}_\mathsf{at}{}^{\overline{\mathcal{P}}}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}, \mathsf{R}^\mathcal{P}_\subseteq)$
return $(\mathsf{true}, \mathsf{st}_\mathcal{P}, \mathsf{num}^{\overline{\mathcal{P}}})$

prune($\mathsf{st}_\mathcal{P}, \mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{cnt}_\mathsf{at}{}^{\overline{\mathcal{P}}}, \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}, \mathsf{R}^\mathcal{P}_\subseteq$)

$\mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen} \leftarrow \mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen} \cup \{\mathsf{cnt}_\mathsf{at}{}^{\overline{\mathcal{P}}}\}$
$\mathsf{st}_\mathcal{P}.\mathsf{R} \leftarrow \mathsf{st}_\mathcal{P}.\mathsf{R} \setminus \mathsf{R}^\mathcal{P}_\subseteq$
for $i \in \mathsf{S}_\mathsf{at}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$ **do**
  $\mathsf{st}_\mathcal{P}.\mathsf{S} \leftarrow \mathsf{st}_\mathcal{P}.\mathsf{S} \setminus \mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{at}[i]$; $\mathsf{st}_\mathcal{P}.\mathsf{S}_\mathsf{at}[i] \leftarrow \emptyset$

Figure 2.14: Optimised UNF-secure ARC scheme ARC-OP based on a RC scheme RC (Definition 1). The sets S, R and S_ack are as in Figure 2.7. The variable max-num represents the largest num received in an at. The counters cnt_at and $\mathsf{max\text{-}cnt}_\mathsf{at}$ keep track of how many at have been sent and largest cnt_at received in an at, respectively. S_at-Seen is the list of cnt_at of received at since the last sent one; $\mathsf{S}_\mathsf{at}[i]$ contains the content of S sent in the ith authentication tag at.

by removing elements in $\mathsf{st}_{\mathcal{P}}[i]$ for $i \in \mathsf{S_{at}\text{-}Seen}^{\overline{\mathcal{P}}}$. The set $\mathsf{S_{at}\text{-}Seen}^{\overline{\mathcal{P}}}$ contains counters of the tags that $\mathcal{P}$ sent to $\overline{\mathcal{P}}$ and $\overline{\mathcal{P}}$ successfully received. Moreover, AuthReceive updates $\mathsf{st}.\mathsf{S_{at}\text{-}Seen}^{\overline{\mathcal{P}}}$ at line 1, after the integrity checks. Since the AuthSend stores the set of sent messages S authenticated within the current at into the array $\mathsf{S_{at}}$, pruning $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ only removes messages that have already been received and authenticated by $\overline{\mathcal{P}}$.

The adversary can only delete and replay authentication tags in the out-of-band channel. We informally discuss how the scheme handles these cases. Assume $\mathcal{P}$ and $\overline{\mathcal{P}}$ exchange some messages, $\mathcal{P}$ receives an authentication tag $\mathsf{at}^{\overline{\mathcal{P}}}$ from $\overline{\mathcal{P}}$ and then sends the authentication tag $\mathsf{at}^{\mathcal{P}}$; the adversary removes $\mathsf{at}^{\mathcal{P}}$ from the channel. This implies that $\mathcal{P}$ does not acknowledge the reception of $\mathsf{at}^{\overline{\mathcal{P}}}$ to $\overline{\mathcal{P}}$ (because AuthSend empties $\mathsf{st}_{\mathcal{P}}.\mathsf{S_{at}\text{-}Seen}$ at every invocation). Consequently, $\overline{\mathcal{P}}$ does not prune $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S}$: these messages will be authenticated with the next authentication tag and security is preserved. The AuthReceive procedure handles adversarial reordering of authentication tags with counters $\mathsf{cnt_{at}}$ at line 5.

Formally, we state the security of ARC-OP in the next theorem.

Theorem 16. Let H be a (t_{cr}, ϵ_{cr})-collision resistant hash function (Definition 6). Then the ARC-OP scheme (Figure 2.14) is correct, ORDINALS secure, and (q, t, ϵ_{cr})-UNF secure, where $t \approx t_{cr}$.

Proof. Correctness and ORDINALS-security for the transformation of Figure 2.14 follow from the scheme in Figure 2.10.

The scheme identical to that in Figure 2.10, modulo the optimizations we introduced. Consequently, the proof of the theorem reduces to showing that the optimizations preserve the security properties of the unoptimized scheme in Figure 2.10. Observe that $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ and $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ are used to detect active attacks. We begin by showing that pruning these sets does not compromise UNF security.

- The set $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ is pruned by removing elements from $\mathsf{R}^{\mathcal{P}}_{\subseteq}$, which was authenticated on line 11 of AuthReceive. We therefore know that messages in $\mathsf{R}^{\mathcal{P}}_{\subseteq}$ are honest. Thus, we can stop sending them to $\overline{\mathcal{P}}$ hereafter.
- $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ is pruned by all sets $\mathsf{st}_{\mathcal{P}}.\mathsf{S_{at}}[i]$ for $i \in \mathsf{S_{at}\text{-}Seen}^{\overline{\mathcal{P}}}$. By construction we know that $\mathsf{S_{at}\text{-}Seen}^{\overline{\mathcal{P}}}$ contains counters for which $\overline{\mathcal{P}}$ *accepted* the authentication tags, since those are included in line 1 of prune. Therefore, $\{\mathsf{st}_{\mathcal{P}}.\mathsf{S_{at}}[i]\}_{i \in \mathsf{S_{at}\text{-}Seen}^{\overline{\mathcal{P}}}}$ contains all messages in $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ that $\overline{\mathcal{P}}$ correctly received and authenticated, which the procedure stores in $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S_{ack}}$ for future checks. Hence, $\mathcal{P}$ can safely stop sending those and prune S correspondingly.

We proceed by showing that UNF security still holds. By the arguments above, an authentication tag at that the AuthSend procedure generates *after* another authentication

tag at', will contain only messages that have *not* been authenticated in at'. Therefore the check on line 5 preserves security.

The check on line 9 verifies whether $\mathsf{R}^{\overline{\mathcal{P}}} \subseteq \mathsf{S}$. Without pruning, this property is met in the absence of forgeries as shown for ARC. Assume for contradiction that $\mathsf{R}^{\overline{\mathcal{P}}}$ contains a message not authenticated yet, but S does not contain this message due to pruning. This means that the message was removed from S by removing one of the values in $\mathsf{st}_{\mathcal{P}}.\mathsf{S}_{\mathsf{at}}$ whose counter $\mathsf{cnt}_{\mathsf{at}}$ was present in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$. Since the counter is present in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$, we know that $\overline{\mathcal{P}}$ accepted the authentication tag containing $\mathsf{cnt}_{\mathsf{at}}$, i.e., $\overline{\mathcal{P}}$ correctly received and authenticated the message. But this means by construction that $\overline{\mathcal{P}}$ pruned the message from R on line 2 of prune, which leads to a contradiction. Therefore the check preserves UNF security.

The check on line 11 verifies whether $\mathsf{R}^{\mathcal{P}}_{\subseteq} \subseteq \mathsf{S}^{\overline{\mathcal{P}}}$. Without pruning, this property is met in absence of forgeries as shown for ARC. Note that $\mathcal{P}$ removes from R only messages that have been authenticated (on line 2 of prune), therefore $\mathsf{R}^{\mathcal{P}}_{\subseteq}$ only contains unauthenticated messages. Similarly, by the argument presented in the paragraph above, $\mathsf{S}^{\overline{\mathcal{P}}}$ contains messages included in at's whose counter was not included in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\mathcal{P}}$, and therefore unauthenticated messages. We conclude that this check also preserves UNF security. □

ARC-OP (Figure 2.14) sends all the authentication material through the out-of-band channel This might be impractical when the authenticated out-of-band channel is narrowband, e.g., if parties use QR-codes to authenticate the communication. We can improve the scheme by using both channels: use the insecure channel to send the authentication data and the possibly narrowband authenticated channel to verify the integrity of those data [BSSW02]. While the idea of using both channels for authentication is natural, some security risks might arise when the scheme does not correctly match the two messages. Since UNF-security depends on both the messages, and therefore on the messages being correctly matched, it might be safer to enforce this property at the scheme level. In the full version of this the work on which this chapter builds [BCC+23b, Appendix G] we propose BW-UNFORGEABLE, a security game that enforces matching of the two authentication messages at the scheme level.

2.5.5 Lightweight bidirectional authentication

We propose a three-move bidirectional authentication protocol, outlined at a high level in Figure 2.15. In this protocol, parties include only their set of *received* messages in the authentication tag. The receiver then compares their counterpart's set of received messages with their own sent messages. This approach suits situations where participants meet in person or online and can simultaneously authenticate each other's view of the conversation, as required in Signal (safety numbers) and other services.

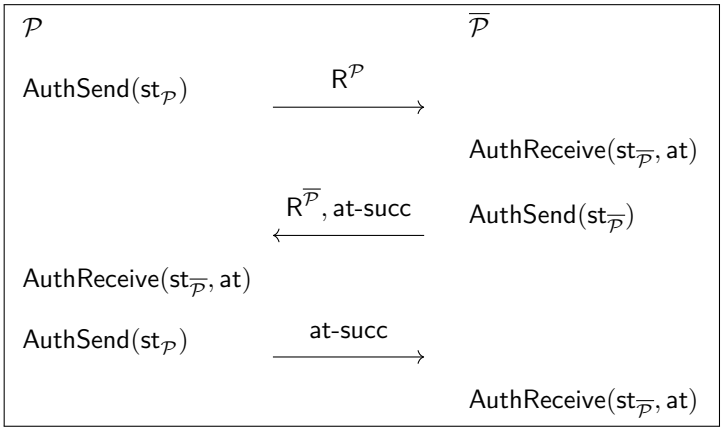

Figure 2.15: Description of the three-move authentication procedure. The boolean at-succ indicates whether the counterpart's set of received messages is a subset of the local set of sent messages.

Security model

We modify the UNF game (Section 2.4) by requiring the adversary to run authentication sessions in sequence. We present the corresponding game in Figure 2.16.

When the adversary initiates the authentication protocol with $\mathcal{P}$ as initial sender via the the AUTHSEND′ oracle, the adversary's access to AUTH∗, SEND′ and RECEIVE′ oracles is restricted until the three-move authentication session completes between $\mathcal{P}$ and $\overline{\mathcal{P}}$. This restriction is managed by the next-oob-op variable and is primarily for clarity; in practice, it may not be necessary to limit SEND′ and RECEIVE′ calls. However, participants who authenticate in person would generally not send messages during this time. To handle this, parties can buffer messages during authentication, to be authenticated in the following session. However, buffering messages implies that an attack carried out during the out-of-band authentication will not be detected until the next authentication protocol. For this reason we block in-band communication during the three-move protocol and we encourage this restriction to be maintained also in practice. Moreover, parties are guaranteed slightly weaker security than in the UNF game. Namely, after receiving the first authentication message, the receiver can deduce that their counterpart has not received a forgery but not that they themselves have until they receive the third message in the protocol.

In Definition 17 we define 3M-UNFORGEABLE-security for ARC schemes.

Definition 17 (3M-UNFORGEABLE)**.** Consider the 3M-UNFORGEABLE game of Figure 2.16. We say that an ARC scheme is (q, t, ϵ)-3M-UNFORGEABLE secure if, for all adversaries $\mathcal{A}$ which make at most q oracle queries, and which run in time at most t, we have:

$$\Pr[\text{3M-UNFORGEABLE}^{\mathcal{A}}(1^{\lambda}) \Rightarrow 1] \leq \epsilon.$$

Game 3M-UNFORGEABLE$^{\mathcal{A}}(1^\lambda)$

1 : auth-state$[\cdot] \leftarrow 0$; next-oob-op $\leftarrow \perp$
2 : play UNF with $\mathcal{A}^{\mathcal{O}}(z)$

Oracle SEND$'(\mathcal{P}, \mathsf{ad}, \mathsf{pt}, \mathsf{r})$

1 : **if** auth-state$[\mathcal{P}] \neq 0$ **then return** $\perp$
2 : **return** SEND$(\mathcal{P}, \mathsf{ad}, \mathsf{pt}, \mathsf{r})$

Oracle RECEIVE$'(\mathcal{P}, \mathsf{ad}, \mathsf{ct})$

1 : **if** auth-state$[\mathcal{P}] \neq 0$ **then return** $\perp$
2 : **return** RECEIVE$(\mathcal{P}, \mathsf{ad}, \mathsf{ct})$

Oracle AUTHSEND$'(\mathcal{P})$

1 : **if** next-oob-op $\notin \{(\mathcal{P}, \text{“authsend”}), \perp\}$ **then**
2 : **return** $\perp$
3 : $i \leftarrow i + 1$
4 : $(\mathsf{st}_{\mathcal{P}}, \mathsf{num}, \mathsf{at}) \leftarrow$ AuthSend$(\mathsf{st}_{\mathcal{P}})$
5 : auth$[(\mathcal{P}, i)] \leftarrow \mathsf{at}$; state$[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
6 : init $\leftarrow \mathbb{1}\{$auth-state$[\mathcal{P}] = 0\}$
7 : auth-state$[\mathcal{P}] \leftarrow$ auth-state$[\mathcal{P}] + 1 \bmod 3$
8 : $\log[i] \leftarrow (\text{“authsend”}, \mathcal{P}, \mathsf{num}, \mathsf{at},$ init$)$
9 : next-oob-op $\leftarrow (\overline{\mathcal{P}}, \text{“authrec”}, i)$
10 : **return** $(\mathsf{at}, \mathsf{num})$

forgery$(\log, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x)$

1 : **return** $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \log \ \wedge$
2 : $(\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \log[x]$

Oracle AUTHRECEIVE$'(\mathcal{P}, j)$

1 : **if** next-oob-op $\neq (\mathcal{P}, \text{“authrec”}, j)$ **then**
2 : **return** $\perp$
3 : at $\leftarrow$ auth$[(\overline{\mathcal{P}}, j)]$
4 : **if** at $= \perp$ **then return** $\perp$
5 : $(\mathsf{auth}, \mathsf{st}, \mathsf{num}) \leftarrow$ AuthReceive$(\mathsf{st}_{\mathcal{P}}, \mathsf{at})$
6 : **if** $\neg$auth **then return** $\perp$
7 : $i \leftarrow i + 1$
8 : auth-state$[\mathcal{P}] \leftarrow$ auth-state$[\mathcal{P}] + 1 \bmod 3$
9 : **if** auth-state$[\mathcal{P}] = 0 \wedge$
10 : auth-state$[\overline{\mathcal{P}}] = 0$ **then**
11 : next-oob-op $\leftarrow \perp$
12 : **else**
13 : next-oob-op $\leftarrow (\mathcal{P}, \text{“authsend”})$
14 : $\mathsf{st}_{\mathcal{P}} \leftarrow \mathsf{st}$; state$[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
15 : $\log[i] \leftarrow (\text{“authrec”}, \mathcal{P}, \mathsf{num}, \mathsf{at})$
16 : **return** num

bad-P$(\log, \mathcal{P}, \mathsf{num}', \mathsf{at}, x, y)$

1 : **return** $(y > x) \wedge$
2 : $(\text{“authsend”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{at}, \cdot) = \log[y] \wedge$
3 : $(\text{“authrec”}, \mathcal{P}, \mathsf{num}', \mathsf{at}) \in \log$

bad-$\overline{\text{P}}(\log, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{at})$

1 : **return** num $\leq$ num$' \wedge$
2 : $(\text{“authrec”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{at}) \in \log \ \wedge$
3 : $(\text{“authsend”}, \mathcal{P}, \mathsf{num}', \mathsf{at}, \mathsf{false}) \in \log$

Figure 2.16: 3M-UNFORGEABLE game for $\mathcal{O} = \{\mathsf{SEND}', \mathsf{RECEIVE}', \mathsf{AUTHSEND}', \mathsf{AUTHRECEIVE}', \mathsf{EXP}_{\mathsf{pt}}, \mathsf{EXP}_{\mathsf{st}}\}$. Highlighted statements correspond to differences relative to the UNF game (Figure 2.8).

The oracles in Figure 2.16 require participants to send all authentication tags via the out-of-band channel. By not providing security guarantees for the first authentication tag, it is possible to send the first tag over the in-band channel and later authenticate it in the second message with an additional hash [PV06]. This approach makes the protocol nearly non-interactive out-of-band: the initiator sends the first authentication tag via the in-band channel and the counterpart responds with the authentication tag and hash out-of-band, such as through a QR code. The final $\mathsf{at\text{-}succ}$ bit can then be determined based on the QR code scanning success or failure. In contrast, solutions like Signal's safety numbers [Mar17b] require *both* parties to scan QR codes out-of-band.

Scheme description

We present a 3M-UNFORGEABLE-secure scheme in Figure 2.17. The $\mathsf{AuthSend}$ and $\mathsf{AuthReceive}$ procedures encode the three-move authentication protocol of Figure 2.15. To identify the different states of the bidirectional authentication, we borrow the terminology from TCP and refer to SYN, $\mathsf{SYN\text{-}ACK}$, and ACK messages and roles. When a party $\mathcal{P}$ first calls $\mathsf{AuthSend}$, it takes the SYN role and sends to $\overline{\mathcal{P}}$ the set of *received* messages and the current num, i.e., $\mathsf{at} \leftarrow (\mathsf{R}, \mathsf{num})$; this set is stored in a separate set R_{at}. As described below, we use R_{at} in $\mathsf{AuthReceive}$ to optimize the scheme. The counterpart $\overline{\mathcal{P}}$ replies with a $\mathsf{SYN\text{-}ACK}$ message, containing its set of received messages, the current ordinal num and the bit $\mathsf{at\text{-}succ}$. The bit $\mathsf{at\text{-}succ}$ indicates whether $\mathcal{P}$'s set of received messages is included in $\overline{\mathcal{P}}$'s set of sent messages (line 10), i.e., $\mathsf{at\text{-}succ}$ indicates whether the authentication of $\mathcal{P}$'s set of received messages was successful. As the counterpart, $\overline{\mathcal{P}}$ stores the current set of received messages in R_{at}. Upon receiving the $\mathsf{SYN\text{-}ACK}$ message, the initiator $\mathcal{P}$ checks whether $\mathsf{at\text{-}succ}^{\overline{\mathcal{P}}} = \mathsf{true}$ and rejects the authentication tag otherwise. $\mathcal{P}$ then sends the ACK message $\mathsf{at} \leftarrow (\mathsf{num}, \mathsf{at\text{-}succ})$. Finally, $\overline{\mathcal{P}}$ calls $\mathsf{AuthReceive}$ to process the ACK message. The party checks the $\mathsf{at\text{-}succ}$ variable to verify that the set $\mathsf{R}^{\overline{\mathcal{P}}}$ is a subset of $\mathsf{S}^{\mathcal{P}}$. If the check passes, the authentication protocol ends.

The optimization of the scheme consists in pruning the set of received messages as soon as the counterpart authenticates them. This reduces the size of the authentication tags, since parties include in at only the received messages that have not been authenticated yet. The $\mathsf{AuthReceive}$ algorithm on line 11 checks whether the counterpart authenticated set of received messages R. If this is the case, all the authenticated messages are stored in $\mathsf{R}_{\mathsf{ack}}$—this set is used in the $\mathsf{Receive}$ algorithm to avoid replay attacks—and at the same time those messages are removed from R thanks to set R_{at}, thereby reducing the size of the next authentication tag and memory consumption. After the pruning, the R set contains only received messages that the counterpart *still* needs to authenticate.

Remark 18. Dowling et al. [DGP22] propose a scheme that is broadly similar to ours. In particular, their protocol uses three moves in-band to allow parties to agree on a common set of respectively received messages R and R'. Then, to authenticate messages and detect

3M-ARC.Setup(1^λ)

1 : // As in Figure 2.10
2 : **return** ARC.Setup(1^λ)

3M-ARC.Init(pp)

1 : $(\mathsf{pp}_0, \mathsf{hk}) \leftarrow \mathsf{pp}$
2 : $(\mathsf{st}'_A, \mathsf{st}'_B, z) \leftarrow\!\!\$\ \mathsf{RC.Init}(\mathsf{pp}_0)$
3 : $\mathsf{num} \leftarrow \bot$
4 : $\mathsf{S}, \mathsf{R}, \mathsf{R}_{\mathsf{ack}}, \mathsf{R}_{\mathsf{at}} \leftarrow \emptyset$
5 : $\mathsf{role\text{-}at}, \mathsf{at\text{-}succ} \leftarrow \bot$
6 : $\mathsf{st}_A \leftarrow (\mathsf{st}'_A, \mathsf{hk}, \mathsf{S}, \mathsf{R}, \mathsf{num}, \mathsf{role\text{-}at},$
7 : $\quad \mathsf{at\text{-}succ}, \mathsf{R}_{\mathsf{ack}}, \mathsf{R}_{\mathsf{at}})$
8 : $\mathsf{st}_B \leftarrow (\mathsf{st}'_B, \mathsf{hk}, \mathsf{S}, \mathsf{R}, \mathsf{num}, \mathsf{role\text{-}at},$
9 : $\quad \mathsf{at\text{-}succ}, \mathsf{R}_{\mathsf{ack}}, \mathsf{R}_{\mathsf{at}})$
10 : $z \leftarrow (z', \mathsf{pp})$
11 : **return** $(\mathsf{st}_A, \mathsf{st}_B, z)$

3M-ARC.Send($\mathsf{st}_\mathcal{P}$, ad, pt)

1 : // As in Figure 2.10
2 : **return** ARC.Send($\mathsf{st}_\mathcal{P}$, ad, pt)

3M-ARC.AuthSend($\mathsf{st}_\mathcal{P}$)

1 : $(\cdot, \cdot, \cdot, \mathsf{R}, \mathsf{num}, \mathsf{role\text{-}at}, \mathsf{at\text{-}succ}, \cdot, \mathsf{R}_{\mathsf{at}}) \leftarrow \mathsf{st}_\mathcal{P}$
2 : **if** $\mathsf{role\text{-}at} = \bot$ **then**
3 : $\quad \mathsf{st}_\mathcal{P}.\mathsf{role\text{-}at} \leftarrow \mathsf{SYN}$
4 : $\quad \mathsf{at} \leftarrow (\mathsf{R}, \mathsf{num});\ \mathsf{st}_\mathcal{P}.\mathsf{R}_{\mathsf{at}} \leftarrow \mathsf{R}$
5 : **elseif** $\mathsf{role\text{-}at} = \mathsf{SYN\text{-}ACK}$ **then**
6 : $\quad \mathsf{at} \leftarrow (\mathsf{R}, \mathsf{num}, \mathsf{at\text{-}succ})$
7 : $\quad \mathsf{st}_\mathcal{P}.\mathsf{R}_{\mathsf{at}} \leftarrow \mathsf{R};\ \mathsf{st}_\mathcal{P}.\mathsf{at\text{-}succ} \leftarrow \bot$
8 : **else** // $\mathsf{role\text{-}at} = \mathsf{ACK}$
9 : $\quad \mathsf{at} \leftarrow (\mathsf{num}, \mathsf{at\text{-}succ})$
10 : $\quad \mathsf{st}_\mathcal{P}.\mathsf{role\text{-}at}, \mathsf{st}_\mathcal{P}.\mathsf{at\text{-}succ} \leftarrow \bot$
11 : **return** $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{at})$

3M-ARC.Receive($\mathsf{st}_\mathcal{P}$, ad, ct)

1 : $(\mathsf{st}_\mathcal{P}, \mathsf{hk}, \cdot, \mathsf{R}, \cdot, \cdot, \cdot, \mathsf{R}_{\mathsf{ack}}, \cdot) \leftarrow \mathsf{st}_\mathcal{P}$
2 : $(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{RC.Receive}(\mathsf{st}'_\mathcal{P}, \mathsf{ad}, \mathsf{ct})$
3 : **if** $\neg\mathsf{acc}$ **then return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
4 : $h \leftarrow \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$
5 : **if** $\exists\ h' : (\mathsf{num}, h') \in \mathsf{R}_{\mathsf{ack}} \wedge h \neq h'$ **then**
6 : $\quad$ **return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \bot, \bot)$
7 : $\mathsf{R} \leftarrow \mathsf{R} \cup \{(\mathsf{num}, h)\}$
8 : $\mathsf{st}_\mathcal{P} \leftarrow (\mathsf{st}_\mathcal{P}, \mathsf{hk}, \cdot, \mathsf{R}, \cdot, \cdot, \cdot, \mathsf{R}_{\mathsf{ack}}, \cdot)$
9 : **return** $(\mathsf{acc}, \mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{pt})$

3M-ARC.AuthReceive($\mathsf{st}_\mathcal{P}$, at)

1 : $(\cdot, \cdot, \mathsf{S}, \mathsf{R}, \cdot, \mathsf{role\text{-}at},$
2 : $\quad \mathsf{at\text{-}succ}, \mathsf{R}_{\mathsf{ack}}, \mathsf{R}_{\mathsf{at}}) \leftarrow \mathsf{st}_\mathcal{P}$
3 : $\mathsf{R}^{\overline{\mathcal{P}}} \leftarrow \emptyset;\ \mathsf{at\text{-}succ}^{\overline{\mathcal{P}}} \leftarrow \mathsf{true}$
4 : **if** $\mathsf{role\text{-}at} = \bot$ **then**
5 : $\quad \mathsf{role\text{-}at} \leftarrow \mathsf{SYN\text{-}ACK};\ (\mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{num}^{\overline{\mathcal{P}}}) \leftarrow \mathsf{at}$
6 : **elseif** $\mathsf{role\text{-}at} = \mathsf{SYN}$ **then**
7 : $\quad (\mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{num}^{\overline{\mathcal{P}}}, \mathsf{at\text{-}succ}^{\overline{\mathcal{P}}}) \leftarrow \mathsf{at}$
8 : **else** // receive ACK case
9 : $\quad (\mathsf{num}^{\overline{\mathcal{P}}}, \mathsf{at\text{-}succ}^{\overline{\mathcal{P}}}) \leftarrow \mathsf{at}$
10 : $\mathsf{at\text{-}succ} \leftarrow (\mathsf{R}^{\overline{\mathcal{P}}} \stackrel{?}{\subseteq} \mathsf{S})$ // Boolean
11 : **if** $\mathsf{at\text{-}succ}^{\overline{\mathcal{P}}}$ **then**
12 : $\quad \mathsf{R}_{\mathsf{ack}} \leftarrow \mathsf{R}_{\mathsf{ack}} \cup \mathsf{R}_{\mathsf{at}};\ \mathsf{R} \leftarrow \mathsf{R} \setminus \mathsf{R}_{\mathsf{at}}$
13 : $\quad \mathsf{R}_{\mathsf{at}} \leftarrow \emptyset$
14 : **else** // failure
15 : $\quad$ **return** $(\mathsf{false}, \mathsf{st}_\mathcal{P}, \mathsf{num})$
16 : $\mathsf{st}_\mathcal{P} \leftarrow (\cdot, \cdot, \mathsf{S}, \mathsf{R}, \mathsf{num}, \mathsf{role\text{-}at},$
17 : $\quad \mathsf{at\text{-}succ}, \mathsf{R}_{\mathsf{ack}}, \mathsf{R}_{\mathsf{at}})$
18 : **return** $(\mathsf{at\text{-}succ}, \mathsf{st}_\mathcal{P}, \mathsf{num}^{\overline{\mathcal{P}}})$

Figure 2.17: Optimised 3M-UNFORGEABLE-secure ARC scheme 3M-ARC based on a RC scheme RC (Definition 1). ARC refers to the unoptimised ARC defined in Figure 2.10. We assume ARC.Send updates the local variable num. As before, the representation of R communicated can be optimised to contain only a single hash.

active attacks, parties compare a hash $\mathsf{H}(\mathsf{R}, \mathsf{R}')$ for hash function H out-of-band. Note however that they do not consider RID security and that they do not formally treat out-of-order message delivery.

Security analysis

In this section we analyze the security of the 3M-ARC scheme, which we introduce in Figure 2.17. Correctness of 3M-ARC follows form the correctness of the underlying RC scheme. Similarly, ORDINALS security is inherited from the underlying RC scheme. As usual, we argue that 3M-UNFORGEABLE security follows from the collision resistance of the underlying hash function.

Theorem 19 (Unforgeability of 3M-ARC)**.** Let H be a (t_{cr}, ϵ_{cr})-collision resistant hash function (Definition 6). Then the 3M-ARC scheme, that we present in Figure 2.17 is (q, t, ϵ_{cr})-3M-UNFORGEABLE secure ARC scheme where $t \approx t_{cr}$.

Proof. We proceed similarly to the proof of Theorem 13. Without loss of generality, we analyze the authentication of $\mathcal{P}$, who we assume to be the initiator, towards $\overline{\mathcal{P}}$. The adversary cannot call the SEND' and $\mathsf{RECEIVE}'$ oracles once the authentication process is started, therefore the sets S and R of both parties are fixed until the completion of the protocol.

To authenticate the set of received messages $\mathsf{R}^{\mathcal{P}}$, $\mathcal{P}$ first sends $\mathsf{at} \leftarrow (\mathsf{R}^{\mathcal{P}}, \mathsf{num})$ to $\overline{\mathcal{P}}$. To verify the authenticity of $\mathsf{R}^{\mathcal{P}}$, the party $\overline{\mathcal{P}}$ verifies whether $\mathsf{R}^{\mathcal{P}} \subseteq \mathsf{S}^{\overline{\mathcal{P}}}$. By the arguments of the proof for Theorem 13, this reduces to the collision-resistance of the hash function H where the reduction runs in time $t \approx t_{cr}$. After receiving the first tag, $\overline{\mathcal{P}}$ is able to detect forgeries received by $\mathcal{P}$ but not by itself. This is taken into account in the 3M-UNFORGEABLE game (line 3 in $\mathsf{bad}\text{-}\overline{\mathsf{P}}$), which states that a forgery received by $\overline{\mathcal{P}}$ is valid only if it is not detected after receiving the second or third tag in the authentication process. Then, when receiving the second tag $(\mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{num}^{\overline{\mathcal{P}}}, \mathsf{at\text{-}succ}^{\overline{\mathcal{P}}})$ from $\overline{\mathcal{P}}$, $\mathcal{P}$ is able to tell if itself received a forgery if the $\mathsf{at\text{-}succ}^{\overline{\mathcal{P}}} = \mathsf{false}$. By the same arguments as before, $\mathcal{P}$ can tell whether $\overline{\mathcal{P}}$ received a forgery by checking $\mathsf{R}^{\overline{\mathcal{P}}} \subseteq \mathsf{S}^{P}$. Finally, upon receiving the third tag, $\overline{\mathcal{P}}$ can detect a forgery using $\mathsf{at\text{-}succ}^{\mathcal{P}}$.

The optimization maintains 3M-UNFORGEABLE-security. Recall that the goal of the optimization (lines 11-13 in Figure 2.17) is to reduce the size of the R set by storing authenticated messages in $\mathsf{R}_{\mathsf{ack}}$. To achieve this reduction, the party executing AuthReceive removes from R the set R_{at}, which is the set of received messages authenticated by the counterpart through the at-succ variable. Since by construction all the messages in R_{at} have been already authenticated by the counterpart, removing them from R does not remove unauthenticated messages from R. □

2.6 Related work

A growing body of research explores the performance and security of messaging in both two-party [BSJ+17, PR18b, JS18, DV19, CDV21, BRV20] and group settings [ACJM20, ACDT21, AJM22]. Some works offer similar [JS18] and sometimes weaker [JMM19] guarantees for in-band active attack detection, assuming in-order communication. To our knowledge, in-band active attack detection has not been explicitly explored in group messaging. However, schemes like the Messaging Layer Security (MLS) [ACDT21, AJM22] ensure that if the state of two parties is forked, their states become incompatible, in some protocol-specific sense.

Naor et al. [NRS20] introduced the concept of immediate key delivery for key exchange: if one goes offline, the remaining ones should be able to complete it successfully by returning a shared secret. This property is orthogonal to immediate decryption as it focuses on keys instead of messages.

Immediate decryption was first formalized by Alwen et al. [ACD19]. Pijnenburg and Poettering [PP22] recently extend the classic ratcheting paradigm by taking into account the progression of physical time, thereby formalizing ciphertext expiration in order to reduce the negative impacts of immediate decryption on forward security.

Apart from Durak and Vaudenay and Caforio et al. who introduced the RECOVER notions, Dowling et al. [DHRR22] provide r-RECOVER, but not s-RECOVER security via signatures, while providing anonymity guarantees even upon state exposure. Dowling et al. [DGP22] frame their authentication guarantees as follows: if no long-term keys are compromised, then all messages exchanged are authentic. Otherwise, active attacks can be detected out-of-band. They achieve this by signing all messages with long-term keys. Our protocols and security notions can be adapted to achieve these guarantees. In distributed computing, the problem is formalised in terms of accountability, which enables parties to detect faulty (Byzantine) nodes [HKD07, CGG+22]. In multi-party computation, a line of work has explored security with *identifiable abort* [IOZ14] which ensures that if parties fail to compute a given function, they can identify the party that caused the failure.

In more applied work, Milner et al. [MCYR17] consider in-band detection of secret misuse in a more general setting but achieve weaker security guarantees than e.g. RECOVER security [CDV21]. In CONIKS [MBB+15] and a subsequent improvement like SEEMless [CDGM19], users can audit a PKI through a tag that is assumed to be gossiped in an authenticated manner by parties, using ideas from transparency log systems.

2.7 Conclusion

This chapter explores active attack detection for secure messaging systems with immediate decryption. We propose both in-band and out-of-band detection mechanisms, addressing scenarios where adversaries can impersonate users and inject messages on parties' behalf. Given the inherent performance limitations of r-RID and r-UNF security [BCC+23a, Section 6], we investigate performance and security trade-offs to ensure that active attack detection is both efficient and practical for real-world applications. Our analysis highlights the importance of balancing detection accuracy with system performance, paving the way for practical implementations that maintain robust security without sacrificing usability.

3 Authenticated private information retrieval

This chapter introduces protocols for authenticated private information retrieval. These schemes enable a client to fetch a record from a remote database server such that (a) the server does not learn which record the client reads, and (b) the client either obtains the "authentic" record or detects server misbehavior and safely aborts. Both properties are crucial for many applications. Standard private-information-retrieval schemes either do not ensure this form of output authenticity, or they require multiple database replicas with an honest majority. In contrast, we offer multi-server schemes that protect security as long as at least one server is honest. Moreover, if the client can obtain a short digest of the database out of band, then our schemes require only a single server. Performing an authenticated private PGP-public-key lookup on an OpenPGP key server's database of 3.5 million keys (3 GiB), using two non-colluding servers, takes under 1.2 core-seconds of computation, essentially matching the time taken by unauthenticated private information retrieval. Our authenticated single-server schemes are 30–100× more costly than state-of-the-art unauthenticated single-server schemes, though they achieve incomparably stronger integrity properties.

An extended abstract of this work appeared at USENIX Security 2023 [CNCG+23a] and the full version is available on the Cryptology ePrint Archive [CNCG+23b]. This chapter's contributions result from a collaboration with Kirill Nikitin, Henry Corrigan-Gibbs, David J. Wu and Bryan Ford. The author of this thesis significantly contributed to defining multi-server and single-server authenticated PIR, designing and proving the security of multi-server schemes, and implementing and evaluating multi-server schemes and Keyd, a PGP key-directory server introduced in this chapter. Kirill Nikitin's PhD thesis [Nik21] also details the single-server authenticated PIR construction based on the decisional Diffie-Hellman assumption (Section 3.5.2), that we include here for completeness.

3.1 Introduction

Private information retrieval (PIR) [CGKS95] enables a client to fetch a record from a database while hiding from the database server(s) which specific record(s) the client retrieves. PIR has numerous privacy-protection uses, such as in metadata-private messaging [AS16, ACLS18], certificate transparency [LG15, Rya14, HHC+23], video streaming [GCM+16], password-breach alerting [TPY+19, LPA+19, ALP+21, PIB+22], private blocklist lookups [KC21], retrieval of software security updates [Cap13], private web search [HDCZ23], public-key directories [Mar17a], and private SQL-like queries on public data [OG10, WYG+17].

Most PIR protocols, however, do not ensure data authenticity in the presence of malicious servers. In many multi-server PIR schemes [CGKS95, BGI16], a single adversarial server can flip any subset of bits in the client's recovered output. In all single-server PIR schemes we know of (c.f., [KO97, CMS99, Lip05, AMBFK16, PPY18, BIPW17, ACLS18, GH19, CK20, PT20, ALP+21, MCR21, MW22, HHCG+23, DPC23] for a non-exhaustive list), a malicious server can choose the exact output that the client will receive by substituting all the database records with a chosen record before processing the client's request. In applications where data integrity matters, such as a PGP public-key directory, unauthenticated PIR is inadequate.

This paper introduces *authenticated private information retrieval*, which augments the standard privacy properties of classic PIR with strong authenticity guarantees. In the multi-server setting, we propose authenticated-PIR schemes for:

- *Point queries*, in which a client wants to fetch a particular database record. For example, "What is the public key for `user@usenix.org`?"
- *Predicate queries*, where a client wants to apply an aggregation operator – such as `COUNT`, `SUM`, or `AVG` – to all records matching a predicate. For example, "How many keys are registered for email addresses ending in `@usenix.org`?"

Our authenticated-PIR schemes guarantee integrity in the *anytrust* model [WCGFJ12]: as long as at least one of the PIR servers is honest. In contrast, prior work that deals with malicious or faulty PIR servers in the multi-server setting either requires a majority or supermajority of servers to be honest [BS02, BS07, Gol07, DGH12] or requires expensive public-key cryptography operations [ZS14]. Our schemes use only fast symmetric-key cryptography in the multi-server setting.

In the single-server setting, we offer authenticated-PIR schemes for point queries which provide authentication as long as the client can obtain a short digest of the database via out-of-band means (Figure 3.1). Prior work for the single-server setting [KO97, WZ18, ZWH21] ensures only that the server truthfully answers the query with respect to *some*

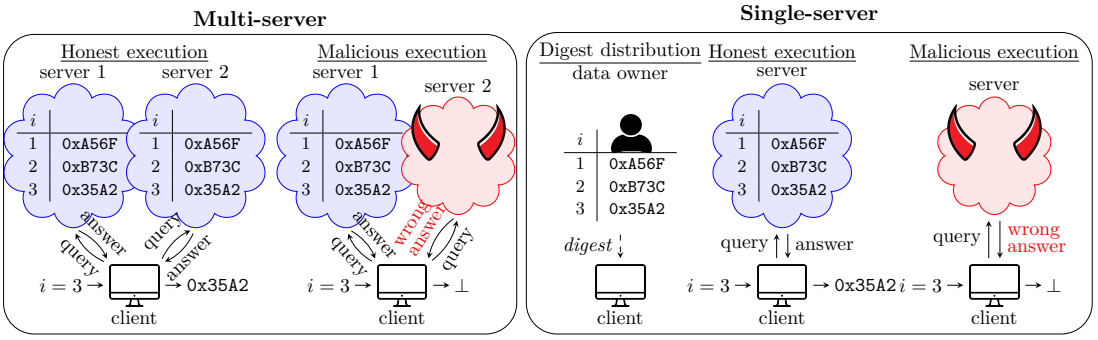

Figure 3.1: In multi-server authenticated PIR, $k \geq 2$ servers hold an exact replica of the database and the client's output is consistent with the honest server's view of the database. If at least one server is honest, the client detects any malicious behaviour from the other servers that reply with respect to an altered database, and rejects the answers. In the single-server setting, a potentially-malicious PIR server holds the database outsourced by the data owner. The client's output is consistent with a database digest that the client obtained from the honest data owner.

database—not necessarily the database the client queried. Table 3.1 summarizes prior work and Section 3.8 gives the complete discussion.

New definitions. Our first contribution is a new definition of integrity for private information retrieval. In our multi-server PIR schemes, a client communicates with several database servers, and client privacy holds as long as at least one server is honest. In this multi-server setting, we say that a PIR scheme satisfies integrity if, whenever the client accepts the servers' answers, the client's output is consistent with an honest server's view of the database.

Defining integrity in the single-server setting is more tricky: If the single database server is malicious, who is to say what the "right" database is? Our approach assumes that the client can obtain a short digest of the database via some out-of-band means. A single-server PIR protocol satisfies integrity if the client accepts the protocol's output only if the output is consistent with the database that the digest represents. In some applications of PIR, the client could obtain this database digest via a gossip mechanism, as in CONIKS [MBB+15], or from a collective authority [STV+16], or from a signature-producing blockchain [NKKJ+17]. In other applications of PIR such as video streaming [GCM+16], a database *owner*—distinct from the PIR servers—might produce, sign, and distribute this digest.

A subtle and important point is that our security definitions require protection against *selective-failure attacks* by malicious servers [KO97, KS06, HKE13]. In this class of attacks, a malicious server answers the client's query with respect to a database that differs from the true database in a few rows. By observing whether the client accepts

or rejects the resulting answer, the server can learn information about which rows the client had queried. To defend against these attacks, our security definitions require that *any* misbehavior on the part of a malicious server causes a client to reject the servers' response.

New constructions. We construct new authenticated-PIR schemes in the multi- and single-server settings.

Multiple servers, point queries. Our first multi-server PIR scheme allows the client to make only *point* queries—to fetch single records from the database. The scheme is simple to implement and has minimal performance overhead. In this scheme, the servers compute a Merkle tree over the database rows and send the client the Merkle root. The client aborts if the servers send different roots. The client then uses unauthenticated PIR to fetch its desired row and a Merkle inclusion proof with respect to the root. The scheme provides authentication when composed with certain—though not all—standard PIR schemes. (Kushilevitz and Ostrovsky suggested using Merkle trees in this setting [KO97], though we are the first to formalize the approach and identify the class of PIR schemes for which it is secure.) On a database containing N records of ℓ bits, and on security parameter λ, our two-server authenticated-PIR scheme for point queries has communication cost $O(\lambda \log N + \ell)$, which matches the cost of the best unauthenticated schemes. Experimentally, this form of authentication imposes less than 3× computational and 1.6× bandwidth overhead, compared with unauthenticated PIR.

Multiple servers, predicate queries. Our multi-server scheme for predicate queries starts with an existing unauthenticated scheme based on function secret sharing [BGI15, BGI16, WYG+17]. We cannot use Merkle trees for authentication: the space of possible queries is exponentially large, so the servers cannot precompute and authenticate each potential answer as before. The client instead uses an information-theoretic message-authentication code—common in malicious secure multiparty protocols [DPSZ12, CDF+08]—to detect whether a server has tampered with its answer. Asymptotically, the communication and computation of our authenticated-PIR scheme for predicate queries matches the costs of the corresponding unauthenticated scheme. Empirically, the authenticated scheme incurs a median overhead of less than 1.02× for both user time and bandwidth. Our multi-server scheme for predicate queries is concretely more computationally expensive (at least 350×) than our scheme for point queries because the cost of evaluating the function secret shares is non-trivial. Thus, this scheme does not scale as well to a large number of servers compared to our specialized multi-server scheme for point queries.

Single server, point queries. Finally, we give two single-server authenticated-PIR protocols: one from the learning-with-errors assumption, and one from the decisional-Diffie-Hellman assumption. Like many recent single-server PIR protocols [AMBFK16, ACLS18, ALP+21, HHCG+23], our schemes extend the classic Kushilevitz-Ostrovsky scheme based on

additively homomorphic encryption [KO97, OS07]. Our schemes incorporate additional randomness that the client uses to authenticate the server's response. The client verifies the server's reply using a short database digest that the client obtains via out-of-band means. Our schemes operate with single-bit records. We propose extensions for handling larger records, but they require increased client computation: more efficient single-server, multi-bit authenticated PIR remains a promising area for future work. Over a database of size N and with security parameter λ, our single-server authenticated-PIR schemes have communication cost $\sqrt{N} \cdot \mathsf{poly}(\lambda)$. In contrast, unauthenticated schemes have communication cost as low as $\log N \cdot \mathsf{poly}(\lambda)$. Our fastest single-server scheme is 30-100× more computationally expensive than the fastest unauthenticated scheme.

An example application. To evaluate authenticated PIR in the context of a practical application, we design and build Keyd, a privacy-preserving PGP public-key directory deployed in the two-server setting. A Keyd client can query the servers for the PGP public key corresponding to a particular email address without leaking the queried email address to the servers. Moreover, a Keyd client can also query the servers for private analysis of the PGP public keys dataset by issuing conjunctive `COUNT`, `SUM` and `AVG` queries without leaking the parameter of the keys over which the predicate is computed. For example, a client can issue a query of the form `SELECT COUNT(*) FROM keys WHERE keyAlgorithm = p`, where p represents the hidden parameter of the predicate, e.g., RSA or ElGamal. Our new authenticated-PIR schemes provide the client with a strong integrity guarantee about the output of the protocols. When run on a recent dump of the SKS PGP key directory, including over 3.5 million keys, querying for a particular key takes the client 1.11 seconds, compared with 1.10 seconds with unauthenticated PIR. Issuing predicate queries with Keyd on the same database imposes an overhead of 1.01× on user time and of 1.05× on bandwidth compared with unauthenticated PIR.

3.1.1 Summary

In summary, in this chapter we make the following contributions:

- In Section 3.3 we introduce the first definition of authenticated PIR in both dishonest-majority and single-server settings. These schemes atomically ensure query privacy and authenticity of the data that the client retrieves.
- In Section 3.4 we present a multi-server authenticated PIR scheme for point queries and another multi-server authenticated PIR scheme for predicate queries, which enables a client to compute a non-trivial function over the database records.
- Section 3.5 introduces two single-server schemes for point queries. The first scheme builds on the decisional Diffie-Hellman assumption and the other on the learning with errors assumption.

PIR scheme	*No. of honest servers needed*	*Malicious*	*Selective-failure secure*	*No public-key cryptography*	*Recovery*
Multi-server schemes					
Robust PIR [BS02, BS07]	1	✗	✗	✓	✓
Byzantine PIR [BS02, BS07, Gol07, DGH12, Kur19]	$>2k/3$	✓	✓	✓	✓
Fault-tolerant PIR [YXB02]	$>k/2$	✓	✓	✓	✓
Verifiable PIR [ZS14]	1	✓	✓	✗	✗
Authenticated PIR (§3.4, §3.5)	1	✓	✓	✓	✗
Single-server schemes					
KO97 [KO97]	0	✓	✗	✗	✗
Verifiable PIR [WZ18, ZWH21]	0	✓	✗	✗	✗
Authenticated PIR (§3.5)	0	✓	✓	✗	✗

Table 3.1: Summary of PIR schemes that tolerate dishonest servers. The multi-server schemes assume a total of k servers. *Malicious* indicates whether the schemes resist malicious adversaries, rather than just faulty servers. *Selective-failure secure* indicates schemes designed to resist selective-failure attacks [KS06]. *No public-key cryptography* refers to schemes that require only fast symmetric primitives; single-server schemes always require public-key operations [CMO00]. *Recovery* indicates whether, in case of a server's misbehaviour, the client is able to recover the correct output or just aborts.

- We implement (Section 3.6) and evaluate (Section 3.7) all the schemes that we propose, assessing the practicality of authenticated PIR with Keyd, a PGP public-key directory service that we build our multi-server authenticated PIR schemes.

3.2 Background and motivation

This section reviews classic PIR schemes, and why naïvely introducing integrity protection into them is unsafe.

3.2.1 Private information retrieval (PIR)

A PIR protocol [CGKS95] takes place between a client and one or more servers. Each server holds a copy of a database consisting of a set of equal-length records. The client wants to query the database without revealing the details of its query to the servers. Modern PIR protocols support two types of queries: (1) the client can fetch a single record from the database, without revealing *which record* it retrieved, or more generally, (2) the client can evaluate a function on all the database records, without revealing *which*

function it evaluated. Non-trivial PIR schemes must also be communication efficient, requiring the client and servers to exchange a number of bits sublinear in the database size. Otherwise, the client could simply download the entire database and perform the query locally.

There are two main types of PIR protocols: multi-server and single-server. In multi-server PIR [CGKS95], the client communicates with $k > 1$ database replicas; correctness holds if all k servers are honest and privacy holds if at least one server is honest. Multi-server PIR schemes traditionally offer information-theoretic privacy. In single-server PIR schemes ($k = 1$) [KO97], correctness holds if the single server is honest and privacy holds against a dishonest server. Single-server PIR schemes require a computationally-bounded server and public-key cryptographic operations [CMO00].

In many applications, the database is a list of (keyword, value) pairs; the PIR client holds a keyword and wants the associated value. In this chapter, we construct authenticated PIR schemes for integer-indexed arrays, and we use off-the-shelf methods [CGN98, GI14] to convert these schemes into authenticated keyword-based PIR schemes.

3.2.2 Why integrity matters in PIR

Standard PIR schemes give the client *no integrity guarantees*. If any one of the servers in a single- or multi-server scheme deviates from the protocol, the malicious server can—in many PIR protocols—completely control the output that the client receives. In other words, classic PIR protocols do not ensure correctness against even just one malicious server.

This lack of integrity protection is extremely problematic in many applications of PIR:

- *Public-key server*: If a client uses PIR to query a PGP or Signal key server for a contact's public keys, a malicious server could cause the client to fetch a false public key for which the adversary controls the secret key.
- *Domain name system*: If a client uses PIR to query a DNS resolver, a malicious PIR server could cause the client to recover the wrong IP address for a hostname and thus poison the client's DNS cache.
- *Online certificate status protocol (OCSP)*: If a client uses PIR to query the revocation status of a public key, a malicious PIR server could trick the client into trusting a certificate that was revoked by the CA after compromise.
- *Content library*: If a client uses PIR to fetch a movie [GCM+16] or a software update, a malicious PIR server could cause the client to recover a malware-infected file instead.

Non-private variants of these applications can already offer integrity. For example, CONIKS [MBB+15] provides integrity of key bindings for public-key directory servers

and DNSSEC [AAL+05] ensures integrity of DNS data. The challenge is thus to ensure integrity in the *private* variants of these applications.

3.2.3 Selective failure and other attacks on PIR

We can always compose standard authentication mechanisms with PIR. For example, a *database owner* – the party responsible for its creation – can append to each database row a digital signature on the record under the database owner's key or a Merkle inclusion proof with respect to a known root. The database owner can then outsource the authenticated database to an untrusted PIR server. After performing a query, the client simply checks the authentication tag on the row it retrieved.

This attempt at authenticated PIR is insecure and vulnerable to *selective-failure attacks* [KS06]. In such attacks, a malicious PIR server selectively corrupts the database so that only targeted queries fail the integrity check. Suppose a malicious PIR server "guesses" that the client is likely to access a particular record, and corrupts *only* that record. The client's integrity check then fails only if the attacker's guess was correct. If the attacker can determine whether the client accepted or rejected the PIR protocol's output—e.g., via the client's subsequent behavior—the attacker can violate client privacy.

Naïve composition can yield other security and privacy hazards. For example, if authentication tags attached to database rows do not uniquely identify the database version and row number, then a malicious PIR server might undetectably swap or duplicate rows or replay old database versions.

Even in a multi-server setting where one malicious server cannot unilaterally corrupt database rows independently, but is limited to blindly flipping bits in its answer without knowing which row these bit-flips will affect, more subtle attacks on naïve compositions may be readily feasible. If rows are protected by malleable digital signatures [DDN91], for example, then a malicious server might flip signature bits in the result so that the signature of a particular "guessed" database row becomes a *different* still-valid signature the client will accept, while the signatures on all other rows become invalid.

3.3 Defining authenticated PIR

We now define *authenticated PIR* in the multi- and single-server settings. In both models, we wish to ensure that the client either obtains "correct" (authentic) output, or else safely rejects the answer without leaking any private information. Privacy must hold even if the PIR servers learn whether the client has accepted or rejected the answer. Therefore, our protocols protect against selective-failure attacks (Section 3.2.3).

3.3.1 Multi-server definition

We now define k-server authenticated PIR schemes, for $k \geq 2$. Our definition generalizes private information retrieval to *weighted functions* of the database rows: the client has a secret function f in mind, which must come from a particular class of functions $\mathcal{F}$. The servers hold a database $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and public "weights" $(w_1, \ldots, w_N)$, one per database row. The client's goal is to get the weighted sum of its private function f applied to each of the rows: $\sum_{i \in [N]} w_i f(i, \mathbf{x}_i)$. When the function class $\mathcal{F}$ is expressive enough, this general syntax subsumes not only the usual definition of multi-server PIR, but also more expressive PIR schemes for predicate queries.

Definition 20 (k-server authenticated PIR for predicate queries)**.** A *k-server authenticated PIR* scheme for function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^\ell, \mathbb{F}]$, database size $N \in \mathbb{N}$, and weights $\mathbf{w} \in \mathbb{F}^N$, consists of three efficient algorithms:

- $\mathsf{Query}(1^\lambda, f) \to (\mathsf{st}, q_1, \ldots, q_k)$. Given a security parameter λ, expressed in unary, and a function $f \in \mathcal{F}$, return secret client state st and queries $q_1, \ldots, q_k$, one per server.
- $\mathsf{Answer}(\mathbf{X}, \mathbf{w}, q) \to a$. Apply query q to database $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\{0,1\}^\ell)^N$ together with weights $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{F}^N$ and return answer a.
- $\mathsf{Reconstruct}(\mathsf{st}, a_1, \ldots, a_k) \to \left\{ \sum_{i \in [N]} w_i f(i, \mathbf{x}_i), \perp \right\}$. Take as input client state st and answers $a_1, \ldots, a_k$ and return the weighted output of the function f applied to the rows of database $\mathbf{X}$, or an error $\perp$.

A k-server authenticated-PIR protocol must satisfy the following properties, which we state formally and informally.

Correctness. Informally, an authenticated-PIR scheme is *correct* if, when an honest client interacts with honest servers, the client always recovers the weighted output of its chosen function applied to the database, i.e., $\sum_{i \in [N]} w_i f(i, \mathbf{x}_i)$.

Definition 21 (Multi-server authenticated PIR correctness)**.** A k-server authenticated-PIR scheme $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ for function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^\ell, \mathbb{F}]$ and database size $N \in \mathbb{N}$ satisfies *correctness* if for every $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$, $\ell \in \mathbb{N}$, $\mathbf{w} \in \mathbb{F}^N$, $\lambda \in \mathbb{N}$, $f \in \mathcal{F}$, the following holds:

$$\Pr\left[\begin{array}{l} y = \sum_{i \in [n]} w_i f(i, \mathbf{x}_i) : \\ \quad (\mathsf{st}, q_1, \ldots, q_k) \leftarrow \mathsf{Query}(1^\lambda, f) \\ \quad\quad a_j \leftarrow \mathsf{Answer}(\mathbf{X}, \mathbf{w}, q_j) \quad \forall j \in [k] \\ \quad\quad y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a_1, \ldots, a_k) \end{array}\right] = 1,$$

where the probability is computed over all the random coins used by the algorithms of the scheme.

Integrity. An authenticated-PIR scheme preserves *integrity with error* ϵ if, when an honest client interacts with a set of k servers, where at most $k-1$ can be malicious and might arbitrarily deviate from the protocol, the client either: outputs the sum of products of its desired function and weights applied to the database, or outputs the error symbol $\perp$, except with probability ϵ. If the scheme has negligible integrity error, we just say that it "preserves integrity." Classic PIR schemes do not ensure this integrity property.

Definition 22 (Multi-server authenticated PIR integrity)**.** A k-server authenticated-PIR scheme $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ for function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^{\ell}, \mathbb{F}]$ and database size $N \in \mathbb{N}$ preserves *integrity with error* ϵ if for every efficient adversary $\mathcal{A}$, and for every $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^{\ell}$, $\ell \in \mathbb{N}$, $\mathbf{w} \in \mathbb{F}^N$, $\lambda \in \mathbb{N}$, $f \in \mathcal{F}$, $j_{\mathsf{good}} \in [k]$, the following probability is negligible in the security parameter λ:

$$\Pr\left[\begin{array}{l} y \notin \left\{ \sum_{i \in [N]} w_i f(i, \mathbf{x}_i), \perp \right\} : \\ \qquad (\mathsf{st}, q_1, \ldots, q_k) \leftarrow \mathsf{Query}(1^{\lambda}, f) \\ \qquad \{a_j\}_{j \neq j_{\mathsf{good}}} \leftarrow \mathcal{A}(\mathbf{X}, \mathbf{w}, \{q_j\}_{j \neq j_{\mathsf{good}}}) \\ \qquad a_{j_{\mathsf{good}}} \leftarrow \mathsf{Answer}(\mathbf{X}, \mathbf{w}, q_{j_{\mathsf{good}}}) \\ \qquad y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a_1, \ldots, a_k) \end{array}\right] \leq \epsilon,$$

where the probability is computed over all the random coins used by the algorithms of the scheme.

Privacy (against malicious servers). An authenticated-PIR scheme satisfies *privacy* if any coalition of up to $k-1$ malicious servers "learns nothing"—in a strong cryptographic sense—about which function in the function class $\mathcal{F}$ the client wants to evaluate on the database, even if the servers learn whether the client's output was the error symbol $\perp$ during reconstruction. Standard PIR schemes do not necessarily satisfy our strong notion of privacy, since such schemes may be vulnerable to selective-failure attacks (Section 3.2.3); authenticated-PIR schemes that provide privacy are not.

Definition 23 (Authenticated PIR privacy)**.** Let $\Pi = (\mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ be a k-server authenticated-PIR scheme for function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^{\ell}, \mathbb{F}]$ and database size $N \in \mathbb{N}$. For $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^{\ell}$, $\ell \in \mathbb{N}$, $\mathbf{w} \in \mathbb{F}^N$, $\lambda \in \mathbb{N}$, $f \in \mathcal{F}$,

$j_{\mathsf{good}} \in [k]$, and an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, define the distribution

$$\mathrm{REAL}_{\mathcal{A},j_{\mathsf{good}},f,\lambda,\mathbf{X},\mathbf{w}} = \left\{ \hat{\beta} : \begin{array}{l} (\mathsf{st}, q_1, \ldots, q_k) \leftarrow \mathsf{Query}(1^\lambda, f) \\ a_{j_{\mathsf{good}}} \leftarrow \mathsf{Answer}(\mathbf{X}, \mathbf{w}, q_{j_{\mathsf{good}}}) \\ \left(\mathsf{st}_{\mathcal{A}}, \{a_j\}_{j \neq j_{\mathsf{good}}}\right) \leftarrow \mathcal{A}_0(\mathbf{X}, \mathbf{w}, \{q_j\}_{j \neq j_{\mathsf{good}}}) \\ y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a_1, \ldots, a_k) \\ b \leftarrow \mathbb{1}\{y \neq \perp\} \\ \hat{\beta} \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\}.$$

Similarly, for a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, define the distribution

$$\mathrm{IDEAL}_{\mathcal{A},\mathcal{S},\mathcal{F},\lambda,\mathbf{X},\mathbf{w}} = \left\{ \beta : \begin{array}{l} (\mathsf{st}_{\mathcal{S}}, Q) \leftarrow \mathcal{S}_0(1^\lambda, \mathcal{F}, \mathbf{X}, \mathbf{w}) \\ (\mathsf{st}_{\mathcal{A}}, A) \leftarrow \mathcal{A}_0(\mathbf{X}, \mathbf{w}, Q) \\ b \leftarrow \mathcal{S}_1(\mathsf{st}_{\mathcal{S}}, A) \\ \beta \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\}.$$

We say Π is *private* if for every efficient adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, and for every $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\{0,1\}^\ell)^N$, $\mathbf{w} \in \mathbb{F}^N$, there exists a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ such that for all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}$, $j_{\mathsf{good}} \in [k]$, the following holds:

$$\mathrm{REAL}_{\mathcal{A},j_{\mathsf{good}},f,\lambda,\mathbf{X},\mathbf{w}} \approx_c \mathrm{IDEAL}_{\mathcal{A},\mathcal{S},\mathcal{F},\lambda,\mathbf{X},\mathbf{w}}$$

We say that an authenticated-PIR scheme is *secure* if it satisfies both integrity and privacy. We define integrity and privacy separately because, as Section 3.3.3 shows, we can reduce the integrity error of a PIR scheme that provides privacy.

Remark 24 (Selective-failure attacks)**.** Including the acceptance bit in the adversary's view ensures protection against selective failure attacks, where whether a client accepts or not leaks information about the client's query. For example, an authenticated-PIR scheme execution, a malicious server could replace a single record i in the database with garbage. Now, if the client's query does not depend on the value of record i, then everything proceeds normally. However, if the query does depend on the value of record i, then it receives a garbage value. Depending on the application, receiving a garbage value could cause the client to abort the protocol prematurely, or retry the protocol; in both of these cases, if the client engages in some kind of recovery mechanism, the server immediately learns information about the client's chosen index i. Definition 23 captures security against selective failure attacks by requiring that the probability of whether the client's response is valid or not (i.e., whether $y \neq \perp$) is *not* correlated with the client's query (since the *same* simulator works for all functions f and moreover, the simulator is not provided f as input). In this way, a malicious server that learns whether the protocol completed successfully or not still cannot learn anything about the client's query.

Example 25 (PIR for point queries—Standard PIR)**.** In authenticated-PIR schemes for point queries, as in a standard PIR scheme, a client privately fetches a single database row. We can recover this functionality from Definition 20, where we take the row length $\ell = 1$ for simplicity. The class of functions $\mathcal{F}$ is the class of point functions $\mathcal{F} = \{f^{(1)}, \ldots, f^{(N)}\} \subseteq \mathsf{Funs}[[N] \times \{0,1\}, \mathbb{F}]$, where $f^{(i)}(i, \cdot) = 1$ and $f^{(i)}(i', \cdot) = 0$ for all $i' \neq i$. The weights are the database entries themselves, i.e., $w_i = x_i \in \{0,1\} \subseteq \mathbb{F}$, for $i \in [N]$.

Example 26 (`COUNT` query)**.** A `COUNT` predicate query counts the database entries satisfying a predicate. A client can count the occurrences of a string $\sigma \in \{0,1\}^\ell$ in a database $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$ using the class of functions $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^\ell, \mathbb{F}]$, where $f(\cdot, \mathbf{x}_i) = 1$ if $\mathbf{x}_i = \sigma$ and $f(\cdot, \mathbf{x}_i) = 0$ otherwise, with constant weights $w_i = 1_\mathbb{F}$, $i \in [N]$.

Remark 27 (Security against $k-1$ malicious servers)**.** The form of authenticated PIR we define above requires security to hold even against coalitions of up to $k-1$ malicious servers. This defines the minimal requirement for multi-server PIR schemes, which do not support complete collusion, and is a model frequently used in anonymous communication systems [WCGFJ12, KLDF, AS16]. In particular, the colluding servers can share their queries with each other and agree on the answers. The protocols that we construct satisfy this strong notion of security. A weaker definition requires security to hold against only adversaries that control a lower threshold $t < k-1$ of the servers. Prior work [BS02, BS07, Gol07] takes $t < k/2$ or $t < k/3$. We discuss these and other related approaches in Section 3.8.

Remediation measures. Authenticated-PIR schemes guarantee security with abort: the client either receives the authentic output or aborts, except with negligible probability. This work does not specify what actions a client could take following an abort, such as identifying the misbehaving server or recovering to continue the protocol. However, we briefly outline potential extensions to incorporate such remediation measures.

In the event of an abort, several strategies can address server misbehavior while balancing privacy and accountability. One approach is to accept the abort, acknowledging that denial-of-service attacks remain possible regardless of the (authentication) PIR protocol. For instance, a server could simply deny having received the client's query. If identifying the misbehaving server is not essential, the client may take no further action. However, if accountability is desired, servers could sign both the query and corresponding answer, enabling the client to publicly present evidence of misbehavior without compromising query privacy.

- *Database replication and verification*: The client can download the entire database, replicate the servers' computations, and identify discrepancies between expected and actual responses. Misbehaving servers can then be held accountable by publishing

the incorrect query and answer (q_i, a_i) pairs, for $i \in [k]$. While this approach incurs a communication overhead *linear* in the database size, it enables the client to blame malicious servers without compromising privacy.

- *Zero-knowledge proofs*: A server could use zero-knowledge proofs to demonstrate that its signed (q_i, a_i) pair, $i \in [k]$, matches the correct computation on the database for a well-formed query. This method allows the server's integrity to be verified without revealing the client's query, maintaining privacy. However, it is likely to incur significant concrete costs.

- *Cut-and-choose protocols* [ZHKS16]: The client could send trap queries specifically crafted to verify server honesty. A cut-and-choose mechanism would detect inconsistencies and identify misbehavior while preserving query privacy. This approach is likely the most efficient in terms of communication and computation overhead. Dietz and Tessaro utilize similar verification queries in their work on fully malicious authenticated PIR [DT24].

These strategies represent potential extensions to authenticated-PIR protocols for addressing server misbehavior while preserving the system's privacy and integrity guarantees. Formalizing and evaluating these remediation measures is left as an open problem for future work.

3.3.2 Single-server definition

This section defines single-server authenticated PIR. One challenge to providing integrity in the single-server setting is that the client has no source of information about the database content other than the server itself. (In the multi-server setting, the honest server acts as a source of "ground truth.") A malicious server can answer the client's query with respect to a database of the server's choosing, and completely control the client's output. We address this problem by introducing a public database digest that cryptographically binds the server to a given database and serves as the ground truth in the scheme. In applications, the client must obtain this digest via out-of-band means, e.g., via gossip, as in CONIKS [MBB+15], or from the database owner if the latter is distinct from the PIR server.

We now give the formal definition of a single-server authenticated-PIR scheme, which differs from the multi-server definition in its use of a digest and in the absence of complex queries. We assume for simplicity that each database record consists of a single bit. The definition generalizes naturally to databases with longer rows.

Definition 28 (Single-server authenticated PIR for point queries)**.** A *single-server authenticated PIR* scheme, for a database of size $N \in \mathbb{N}$, consists of the following algorithms:

- $\mathsf{Digest}(1^\lambda, \mathbf{x}) \to d$. Take a security parameter λ (in unary) and a database $\mathbf{x} \in \{0,1\}^N$ and return a digest d.
- $\mathsf{Query}(d, i) \to (\mathsf{st}, q)$. Take as input a digest d and an index $i \in [N]$ and return a client state st and a query q.
- $\mathsf{Answer}(d, \mathbf{x}, q) \to a$. Apply query q to database $\mathbf{x} \in \{0,1\}^N$ with digest d and return answer a.
- $\mathsf{Reconstruct}(\mathsf{st}, a) \to \{0, 1, \perp\}$. Take as input state st and answer a and return a database bit or an error $\perp$.

A single-server authenticated-PIR scheme must satisfy analogous properties to those in the multi-server setting: correctness, integrity and privacy. If a scheme satisfies both integrity and privacy, we say that the scheme is secure. We now present the formal definitions.

Definition 29 (Single-server authenticated PIR correctness)**.** A single-server authenticated-PIR scheme $(\mathsf{Digest}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ satisfies *correctness* if for every database $x \in \{0,1\}^N$, $i \in [N]$, and $\lambda \in \mathbb{N}$, the following holds:

$$\Pr\left[x_i' = x_i : \begin{array}{r} d \leftarrow \mathsf{Digest}(1^\lambda, x) \\ (\mathsf{st}, q) \leftarrow \mathsf{Query}(d, i) \\ a \leftarrow \mathsf{Answer}(d, x, q) \\ x_i' \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

Definition 30 (Single-server authenticated PIR integrity)**.** A single-server authenticated-PIR scheme $(\mathsf{Digest}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ has integrity error ϵ if for every efficient (non-uniform) adversary $\mathcal{A}$, every database $x \in \{0,1\}^N$, and index $i \in [N]$,

$$\Pr\left[x_i' \notin \{x_i, \perp\} : \begin{array}{r} d \leftarrow \mathsf{Digest}(1^\lambda, x) \\ (\mathsf{st}, q) \leftarrow \mathsf{Query}(d, i) \\ a^* \leftarrow \mathcal{A}(d, x, q) \\ x_i' \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*) \end{array}\right] \leq \epsilon(\lambda) + \mathsf{negl}(\lambda),$$

where the probability is *only* taken over the choice of *query randomness*[1]. We say the scheme provides integrity if it has integrity error 0.

Remark 31 (On non-uniform hardness)**.** As written, Definition 30 requires integrity to hold against *non-uniform* adversaries. This version of the assumption explicitly captures the fact that the probability of an integrity failure is only taken over the randomness of query generation (and *not* the adversary). Thus, a malicious server cannot induce

[1]Note that since the adversary is allowed to take *non-uniform* advice, we can assume without loss of generality that the adversary is deterministic (and incur at most a constant loss in advantage).

correlated integrity failures across multiple independently-generated queries. This property is very useful for our integrity amplification transformation (Appendix B.1). We could also consider a more complex (multi-query) variant of this assumption that applies to both uniform and non-uniform adversaries (and which suffices for the transformation in Appendix B.1). For simplicity of exposition, we opt to give the stronger, but simpler-to-describe non-uniform notion here.

Definition 32 (Single-server authenticated PIR privacy)**.** Let $(\mathsf{Digest}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ be a single-server authenticated-PIR scheme. For a security parameter $\lambda \in \mathbb{N}$, a database $x \in \{0,1\}^N$, an index $i \in [N]$, and an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, define the distribution

$$\mathrm{REAL}_{\mathcal{A},x,i,\lambda} := \left\{ \hat{\beta} : \begin{array}{l} d \leftarrow \mathsf{Digest}(1^\lambda, x) \\ (\mathsf{st}, q) \leftarrow \mathsf{Query}(d, i) \\ (\mathsf{st}_{\mathcal{A}}, a^*) \leftarrow \mathcal{A}_0(d, x, q) \\ x_i' \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*) \\ b \leftarrow \mathbb{1}\{x_i' \neq \perp\} \\ \hat{\beta} \leftarrow \mathcal{A}_1(\mathsf{st}_A, b) \end{array} \right\}.$$

Similarly, for a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, define the distribution

$$\mathrm{IDEAL}_{\mathcal{A},\mathcal{S},x,\lambda} := \left\{ \beta : \begin{array}{l} d \leftarrow \mathsf{Digest}(1^\lambda, x) \\ (\mathsf{st}_{\mathcal{S}}, q) \leftarrow \mathcal{S}_0(d, x) \\ (\mathsf{st}_{\mathcal{A}}, a^*) \leftarrow \mathcal{A}_0(d, x, q) \\ b \leftarrow \mathcal{S}_1(\mathsf{st}_{\mathcal{S}}, a^*) \\ \beta \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\}.$$

An authenticated PIR scheme $(\mathsf{Digest}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ has *privacy* if for every adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ there exists a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ such that for every database length $N = N(\lambda)$, database $x \in \{0,1\}^N$, index $i \in [N]$, the following holds:

$$|\Pr[\textsc{real}_{\mathcal{A},x,i,\lambda} = 1] - \Pr[\textsc{ideal}_{\mathcal{A},\mathcal{S},x,\lambda} = 1]| \leq \mathsf{negl}(\lambda).$$

Remark 33 (Adaptive notions of privacy)**.** We could also consider stronger versions of privacy (Definition 32) where the adversary chooses the query *adaptively* after seeing the digest. In both of our single-server authenticated-PIR constructions (Constructions 3 and 4), the digest is a *deterministic* function of the database, and hence, choosing the query adaptively does not help the adversary. For this reason, we opt to give the (simpler) privacy definition.

Malformed digest. Our schemes guarantee integrity for single-server authenticated PIR only when the client uses an honestly-generated digest. In all applications of single-server PIR that we envision, this security guarantee is sufficient—the client's goal is to check that a (possibly malicious) PIR server's answer is consistent with the (correct) digest that the client has obtained out-of-band from the data owner. Stronger notions of security are possible, however. We could require that even if the digest is generated adversarially, the client is guaranteed to recover output that is consistent with *some* n-bit database. This stronger notion is related to that of simulatable adaptive oblivious transfer [CNS07] and extends to other cryptographic primitives [FS09, JL09].

3.3.3 Integrity amplification

The lattice-based single-server authenticated-PIR schemes that we construct in Section 3.5 have noticeable integrity error $\epsilon = 1/\mathsf{poly}(\lambda)$ for some parameter settings. Here we show how to combine an authenticated-PIR scheme that provides privacy and has integrity error ϵ with any error-correcting code to reduce the integrity error to a negligible quantity, in both the multi- and single-server settings. In particular, we prove the following theorem:

Theorem 34 (Integrity amplification, informal)**.** If Π is an authenticated-PIR scheme with privacy and with integrity error ϵ then, for every $t \in \mathbb{N}$, there is an authenticated-PIR scheme Π' with privacy and with integrity error ϵ^{t+1}, where Π' invokes Π at most $2t+1$ times.

The integrity-amplification construction works as follows:

- The server first encodes each database record with an error-correcting code. Suppose each encoded record is n bits and that the error-correcting code can correct t errors. The server constructs n databases where the j^{th} database contains the j^{th} bit of the codeword for each record.

- To retrieve a record i, the client makes n authenticated PIR queries to obtain the n bits of the codeword encoding record i. Let $y_1, \ldots, y_n$ be the responses. If $y_j = \perp$ for any $j \in [n]$, the client rejects with output $\perp$. Otherwise, the client decodes $y = y_1 \cdots y_n$ to obtain the record.

If the error-correcting code supports decoding codewords with up to t errors and the authenticated-PIR scheme has integrity error ϵ, then the integrity error of this construction is at most ϵ^{t+1}. Specifically, to compromise integrity, the server must corrupt at least $t+1$ bits y_j. Integrity of the underlying scheme ensures that the probability the adversary succeeds in corrupting y_j is at most ϵ. Each query is *independent*, so the server's success probability is now ϵ^{t+1}.

A basic instantiation of this paradigm is to instantiate using the simple repetition code, where the encoding of a bit $b \in \{0,1\}$ simply consists of $2t+1$ copies of b. This basic repetition code corrects up to t errors. The client uses the base authenticated PIR scheme Π $2t+1$ times to fetch each of the $2t+1$ bits of the codeword corresponding to its desired database record. If any of these $2t+1$ runs output $\perp$, the client outputs $\perp$. If none of the $2t+1$ runs output $\perp$, then either: (a) the client recovers at least $t+1$ correct bits of the codeword, in which case the client correctly recovers its desired output bit, or (b) the client recovers an incorrect bit on more than t of the protocol runs, which happens with probability at most ϵ^{t+1}, by the ϵ-integrity of the underlying PIR scheme. Setting $t = \lambda/\epsilon$ then yields a construction with negligible integrity error.

When the database records are longer (e.g., field elements instead of bits), we can use better error-correcting codes with higher rate compared to the basic repetition code. This allows amplifying integrity with fewer repetitions. In Appendix B.1, we formally describe a single-server construction that uses an error-correcting code that supports multi-bit records over any field to amplify the integrity of a single-server authenticated PIR scheme.

3.4 Multi-server authenticated PIR

We give two constructions of multi-server authenticated PIR.

3.4.1 Point queries via Merkle trees

We first present a multi-server authenticated-PIR scheme for *point queries*. This scheme enables a client with a secret index $i \in [N]$ to retrieve the i^{th} record from a database of N records.

A natural way to construct an authenticated-PIR scheme is to combine a standard (unauthenticated) multi-server PIR scheme with a standard integrity-protection mechanism, such as Merkle trees [Mer87]. While this composition is in general *insecure* under our definition, we show that it can be secure with a careful choice of the underlying primitives.

Preliminaries: Merkle tree and classic multi-server PIR

We formally define Merkle tree and classic multi-server PIR schemes together with their security properties.

Definition 35. A *Merkle-tree scheme* $\mathsf{M} = (\mathsf{Digest}, \mathsf{ProveIncludes}, \mathsf{VerifyIncludes})$, which is parametrized by a digest length $\ell_{\mathsf{dig}} \in \mathbb{N}$ and a inclusion proof length $\ell_\pi \in \mathbb{N}$, for a database $\mathbf{x} \in \{0,1\}^N$, $N \in \mathbb{N}$, consists of two possibly randomized algorithms and one deterministic algorithm:

- $\mathsf{Digest}(1^\lambda, \mathbf{x}) \to d$. Given a security parameter λ, expressed in unary, and a database $x \in \{0,1\}^N$, returns a database digest $d \in \{0,1\}^{\ell_{\mathsf{dig}}}$.

- $\mathsf{ProveIncludes}(1^\lambda, \mathbf{x}, i, x_i) \to \{\pi_i, \perp\}$. This deterministic algorithm, on input a security parameter λ expressed in unary, a database $\mathbf{x} \in \{0,1\}^N$, an index $i \in [N]$ and a database record $x_i \in \{0,1\}$, outputs a unique proof $\pi_i \in \{0,1\}^{\ell_\pi}$ if $x_i \in \mathbf{x}$ and $\perp$ otherwise.

- $\mathsf{VerifyIncludes}(d, i, x_i, \pi_i) \to \{0,1\}$. Given a digest $d \in \{0,1\}^{\ell_{\mathsf{dig}}}$, a index $i \in [N]$, a database entry $x_i \in \{0,1\}$ and a proof $\pi_i \in \{0,1\}^{\ell_\pi}$, outputs 1 if π_i proves that the database represented by the digest d contains the record x_i at position i and 0 otherwise.

A Merkle-tree scheme defined in Definition 35 is required to satisfy the following properties.

Definition 36 (Merkle tree correctness)**.** Let $\mathsf{M} = (\mathsf{Digest}, \mathsf{ProveIncludes}, \mathsf{VerifyIncludes})$ be a Merkle-tree scheme as defined in Definition 35, parametrized by a digest length $\ell_{\mathsf{dig}} \in \mathbb{N}$ and a inclusion proof length $\ell_\pi \in \mathbb{N}$, for a database $\mathbf{x} \in \{0,1\}^N$, $N \in \mathbb{N}$. We say that M satisfies *correctness* if, for all $i \in [N]$, the following holds:

$$\Pr\left[b = 1 : \begin{array}{c} d \leftarrow \mathsf{Digest}(\mathbf{x}) \\ \pi \leftarrow \mathsf{ProveIncludes}(\mathbf{x}, i, x_i) \\ b \leftarrow \mathsf{VerifyIncludes}(d, i, x_i, \pi) \end{array}\right] = 1.$$

Definition 37 (Merkle tree uniqueness)**.** Let $\mathsf{M} = (\mathsf{Digest}, \mathsf{ProveIncludes}, \mathsf{VerifyIncludes})$ be a Merkle-tree scheme as defined in Definition 35, parametrized by a digest length $\ell_{\mathsf{dig}} \in \mathbb{N}$ and a inclusion proof length $\ell_\pi \in \mathbb{N}$, for a database $\mathbf{x} \in \{0,1\}^N$, $N \in \mathbb{N}$. Let $\mathcal{A}$ be an efficient adversary. M ensures *uniqueness* if the following holds:

$$\Pr\left[b = b' = 1 : \begin{array}{l} (\mathbf{x}, i, x_i, \pi_i, \pi_i') \leftarrow \mathcal{A}(1^\lambda, N) \\ \textit{if } \pi_i = \pi_i' \textit{ then abort} \\ d \leftarrow \mathsf{Digest}(\mathbf{x}) \\ b \leftarrow \mathsf{VerifyIncludes}(d, i, x_i, \pi_i) \\ b' \leftarrow \mathsf{VerifyIncludes}(d, i, x_i, \pi_i') \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 38 (Merkle tree soundness)**.** Let $\mathsf{M} = (\mathsf{Digest}, \mathsf{ProveIncludes}, \mathsf{VerifyIncludes})$ be a Merkle-tree scheme as defined in Definition 35, parametrized by a digest length $\ell_{\mathsf{dig}} \in \mathbb{N}$ and a inclusion proof length $\ell_\pi \in \mathbb{N}$, for a database $\mathbf{x} \in \{0,1\}^N$, $N \in \mathbb{N}$. Let $\mathcal{A}$

be an efficient adversary. M satisfies *soundness* if the following holds:

$$\Pr\left[b = 1 : \begin{array}{l} (\mathbf{x}, i, x_i^*, \pi_i) \leftarrow \mathcal{A}(1^\lambda, N) \\ \textit{if } x_i = x_i^* \textit{ then abort} \\ d \leftarrow \mathsf{Digest}(\mathbf{x}) \\ b \leftarrow \mathsf{VerifyIncludes}(d, i, x_i^*, \pi_i) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We now define standard k-server unauthenticated-PIR schemes, for $k \geq 2$.

Definition 39 (k-server PIR for point queries)**.** A *k-server unauthenticated-PIR* scheme for point queries parametrized by a database length $N \in \mathbb{N}$, consists of three efficient, and possibly randomized, algorithms:

- $\mathsf{Query}(1^\lambda, i) \to (\mathsf{st}, q_1, \ldots, q_k)$. Given a security parameter λ, expressed in unary, and an index $i \in [N]$, return client state st and queries $q_1, \ldots, q_k$.
- $\mathsf{Answer}(\mathbf{x}, q) \to a$. Apply query q to database $\mathbf{x} \in \{0,1\}^N$ and return answer a.
- $\mathsf{Reconstruct}(\mathsf{st}, a_1, \ldots, a_k) \to x_i$. Take as input client state st and answers $a_1, \ldots, a_k$ and return the i^{th} record of the database x_i.

A k-server unauthenticated-PIR scheme is required to satisfy the following properties.

Definition 40 (PIR correctness)**.** An unauthenticated-PIR scheme $\mathsf{PIR} = (\mathsf{PIR.Query}, \mathsf{PIR.Answer}, \mathsf{PIR.Reconstruct})$, parametrized by a number of servers $k \in \mathbb{N}$ and a database size $N \in \mathbb{N}$ satisfies *correctness* if for every $\mathbf{x} \in \{0,1\}^N$, the following holds:

$$\Pr\left[x_i' = x_i : \begin{array}{ll} (\mathsf{st}, \{q_i\}_{i\in[k]}) \leftarrow \mathsf{PIR.Query}(i) & \\ a_j \leftarrow \mathsf{PIR.Answer}(\mathbf{x}, q_j) & \forall j \in [k] \\ x_i' \leftarrow \mathsf{PIR.Reconstruct}(\mathsf{st}, a_1, \ldots, a_k) & \end{array}\right] = 1,$$

where the probability is computed over all the random coins used by the algorithms of the scheme.

Definition 41 (PIR security)**.** Let $\mathsf{PIR} = (\mathsf{PIR.Query}, \mathsf{PIR.Answer}, \mathsf{PIR.Reconstruct})$ be an unauthenticated-PIR scheme for point queries parametrized by a number of servers $k \in \mathbb{N}$ and a database size $N \in \mathbb{N}$. Let S be any subset of $k-1$ elements from $[k]$. For $i \in [N]$ let the distribution

$$\mathrm{REAL}_i = \left\{ \bigcup_{j\in S} q_j : (\mathsf{st}, q_1, \ldots, q_k) \leftarrow \mathsf{PIR.Query}(i) \right\}.$$

Similarly, for a simulator $\mathcal{S}$, let the distribution

$$\mathrm{IDEAL}_{\mathcal{S}} = \left\{ \{q_j\}_{j\in S} \leftarrow \mathcal{S} \right\}.$$

A classic unauthenticated-PIR scheme $\mathsf{PIR} = (\mathsf{PIR.Query}, \mathsf{PIR.Answer}, \mathsf{PIR.Reconstruct})$ parametrized by a database length $N \in \mathbb{N}$ and a number of servers $k \in \mathbb{N}$ is *secure* if for every $i \in [N]$, the following holds:

$$\mathrm{REAL}_i \approx_c \mathrm{IDEAL}_{\mathcal{S}}.$$

In this work, we consider only *linear* classic PIR schemes. Many standard PIR schemes are linear [CGKS95, GI14, BGI16, CK20].

Definition 42 (Linear PIR)**.** Let $\mathsf{PIR} = (\mathsf{PIR.Query}, \mathsf{PIR.Answer}, \mathsf{PIR.Reconstruct})$ be a classic PIR scheme for point queries parametrized by a number of servers $k \in \mathbb{N}$ and a database size $N \in \mathbb{N}$. We say that PIR is a *linear* PIR scheme if the $\mathsf{Reconstruct}$ algorithm is the sum of the individual severs' answers.

We sketch the construction here and formally present it in Construction 1. This construction uses a standard multi-server PIR scheme in which (a) the client sends a single message to each server and receives a single message in return and (b) client reconstructs its output by summing up (or XORing) the answers from the servers. Many standard PIR schemes have this form [CGKS95, GI14, BGI16, CK20] (see Definition 42).

In these schemes, if any of the servers deviate from the prescribed protocol, the worst they can do is to cause the client to recover the correct output shifted by a constant of the adversarial servers' choosing. Therefore, instead of recovering the message $m \in \{0,1\}^\ell$, the client recovers $m \oplus \Delta$, for some non-zero value $\Delta \in \{0,1\}^\ell$.

Our approach then is to have the servers compute a Merkle tree over the N database entries along with their indices: $\{(1, \mathbf{x}_1), \ldots, (N, \mathbf{x}_N)\}$. Call the root of the tree R. Then for each entry, each server constructs a Merkle proof π_i of inclusion in the tree rooted at R and attaches this proof to each database record. The asymptotic complexity of this preprocessing phase is $\mathcal{O}(N)$; we discuss concrete costs in Section 3.7. Finally, the client and servers run the PIR protocol over the database $\{(1, \mathbf{x}_1, \pi_1), \ldots, (N, \mathbf{x}_N, \pi_N)\}$. Each of the servers also sends the Merkle root R to the client.

The client first checks that it received the same Merkle root R from all of the servers. Since at least one of the servers is honest, this ensures the client receives the *honestly-generated* root. If all the roots match, the client reconstructs the record and verifies the Merkle inclusion proof with respect to R. If a server misbehaves, the client will recover $(i', \mathbf{x}'_i, \pi'_i) = (i, \mathbf{x}_i, \pi_i) \oplus \Delta$ for some non-zero offset Δ. Whenever $\Delta \neq 0$, security of the Merkle proof ensures that π'_i will be an invalid proof of $(i, \mathbf{x}_i)$ with respect to R.

We now present the security proofs for Construction 1, beginning with an overview of the proof strategy, followed by the detailed proofs.

Construction 1 (k-server authenticated PIR for point queries tolerating $k-1$ malicious servers)**.** The construction is parametrized by a number of servers $k \in \mathbb{N}$, a number of database rows $N \in \mathbb{N}$, a row length $\ell \in \mathbb{N}$, a security parameter $\lambda \in \mathbb{N}$, a Merkle-tree scheme M (Definition 35), and a linear PIR scheme PIR (Definition 42). Weights are ignored in this scheme. We represent the database as N binary strings of length ℓ each: $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$. The Query algorithm inputs the security parameter 1^λ and an index $i \in [N]$; $\mathsf{Reconstruct}$ outputs either a vector $\mathbf{x}_i \in \{0,1\}^\ell$ or the rejection symbol $\perp$ (see Example 25 to recover this functionality from Definition 20). The servers execute the first three steps of the Answer procedure only when the database changes; we show the entire procedure for completeness.

$\underline{\mathsf{Query}(1^\lambda, i \in [N]) \to (\mathsf{st}, q_1, \ldots, q_k)}$

1. $(\mathsf{st}_{\mathsf{PIR}}, q_1, \ldots, q_k) \leftarrow \mathsf{PIR.Query}(i)$.
2. Set the state $\mathsf{st} \leftarrow (i, \mathsf{st}_{\mathsf{PIR}})$.
3. Output $(\mathsf{st}, q_1, \ldots, q_k)$.

$\underline{\mathsf{Answer}(\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell, q) \to a}$

1. Compute the digest $\mathsf{root} \leftarrow \mathsf{M.Digest}(\mathbf{X})$.
2. For $j \in [n]$, compute $\pi_i \leftarrow \mathsf{M.ProveIncludes}(\mathbf{X}, j, x_j)$.
3. Enlarge the database with the proofs for all the records as $\mathbf{X}' \leftarrow ((\mathbf{x}_1, \pi_1), \ldots, (\mathbf{x}_N, \pi_N))$.
4. Output $(\mathsf{root}, \mathsf{PIR.Answer}(\mathbf{X}', q))$.

$\underline{\mathsf{Reconstruct}(\mathsf{st}, a_1, \ldots, a_k) \to \left\{\{0,1\}^\ell, \perp\right\}}$

1. Parse the state st as $(i, \mathsf{st}_{\mathsf{PIR}})$.
2. For $j \in [k]$, parse a_k as (root_k, a'_k).
3. If the k roots $\{\mathsf{root}_j\}_{j \in [k]}$ are not all equal, return $\perp$.
4. Run the classic PIR reconstruction algorithm and parse $r_i \leftarrow \mathsf{PIR.Reconstruct}(\mathsf{st}_{\mathsf{PIR}}, a'_1, \ldots, a'_k)$ as $(\mathbf{x}_i, \pi_i)$.
5. If $\mathsf{M.VerifyIncludes}(\mathsf{root}_1, i, \mathbf{x}_i, \pi_i) = \perp$, then output $\perp$. Otherwise output $\mathbf{x}_i$.

Overview of the proof strategy

In this section e give a overview of the strategy that we use to prove integrity and privacy for Construction 1. We describe the construction with $k = 2$ servers, but the same intuition generalizes to $k > 2$ servers. This overview is inspired by a private discussion with Brett Falk, Pratyush Mishra, and Matan Shtepel [SMF24], which pointed out a flaw in the proof of Theorem 45.

Construction 1 uses a *linear* classic PIR scheme (Definition 42), i.e., the $\mathsf{PIR.Reconstruct}$ algorithm is the sum of the individual servers' answers. In other words, by denoting the sum operation with $\oplus$, we can rewrite the reconstruction algorithm as

$$\mathsf{PIR.Reconstruct}(\mathsf{st}_{\mathsf{PIR}}, a'_1, \ldots, a'_k) \to a'_1 \oplus \cdots \oplus a'_k.$$

Construction 1 uses $\mathsf{PIR.Reconstruct}$ in line 4 of the $\mathsf{Reconstruct}$ procedure.

For the sake of this overview, we consider that the servers hold a copy of a two-record database $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2) \in (\{0,1\}^\ell)^2$, for row length $\ell \in \mathbb{N}$. Suppose that server 1 is malicious and that server 2 is honest, and suppose that server 1 mounts a selective-failure attack by replacing the record $\mathbf{x}_2$ with a bogus record $\mathbf{x}_2^* \in \{0,1\}^\ell$, i.e., server 1 uses the bogus database $\mathbf{X}^* = (\mathbf{x}_1, \mathbf{x}_2^*) \in (\{0,1\}^\ell)^2$. After the first three steps of the Answer procedure in Construction 1, the malicious server 1 has the following bogus enlarged database:

$$\mathbf{X}'_1 \leftarrow ((\mathbf{x}_1, \pi_1), (\mathbf{x}_2^*, \pi_2^*));$$

the honest server 2 has the correct enlarged database:

$$\mathbf{X}'_2 \leftarrow ((\mathbf{x}_1, \pi_1), (\mathbf{x}_2, \pi_2)).$$

Assume that server 1 sets a honest Merkle root, i.e., server 1 computes $\mathsf{root} \leftarrow \mathsf{M.Digest}(\mathbf{X})$; otherwise the client immediately rejects (line 3 of the $\mathsf{Reconstruct}$ procedure).

Given an index $i \in \{1, 2\}$, the client computes queries q_1, q_2 using the Query procedure. The two servers compute the answers as follows:

$$a_1 \leftarrow (\mathsf{root}, \mathsf{PIR.Answer}(\mathbf{X}'_1, q_1))$$
$$a_2 \leftarrow (\mathsf{root}, \mathsf{PIR.Answer}(\mathbf{X}'_2, q_2))$$

Since the roots are equal, the client runs the classic PIR reconstruction procedure an gets $r_i \leftarrow \mathsf{PIR.Reconstruct}(\mathsf{st}_{\mathsf{PIR}}, a'_1, a'_2)$. By the linearity of the classic PIR scheme and

by setting $\Delta \leftarrow \mathsf{PIR.Answer}(\mathbf{X}'_1, q_1) \oplus \mathsf{PIR.Answer}(\mathbf{X}'_2, q_1)$, we have

$$\begin{aligned} r_i &= a'_1 \oplus a'_2 = \mathsf{PIR.Answer}(\mathbf{X}'_1, q_1) \oplus \mathsf{PIR.Answer}(\mathbf{X}'_2, q_2) \\ &= \mathsf{PIR.Answer}(\mathbf{X}'_2, q_1) \oplus \Delta \oplus \mathsf{PIR.Answer}(\mathbf{X}'_2, q_2) \\ &= (\mathbf{x}_i, \pi_i) \oplus \Delta, \end{aligned}$$

as $\mathbf{X}'_2$ is the honest enlarged database. By assumption the malicious server 1 feeds PIR.Answer with a bogus database, which implies that $\Delta = \mathsf{PIR.Answer}(\mathbf{X}'_1, q_1) \oplus \mathsf{PIR.Answer}(\mathbf{X}'_2, q_1) \neq 0$ and therefore that $(\mathbf{x}_i, \pi_i) \neq (\mathbf{x}_i, \pi_i) \oplus \Delta$. This in turn implies, by soundness and/or uniqueness of the Merkle-tree scheme (Definition 38 and Definition 37) that the client rejects the answers in step 5 of the Reconstruct procedure of Construction 1.

The crux is that the argument holds *regardless* of whether the client queried for index $i = 1$ or $i = 2$: we do not assume a specific index in the argument that leads to client's rejection. In other words, the client rejects, except with negligible probability, whenever one of the servers replies with respect to a bogus database independently from the index that the client inputs to the Query procedure.

Security proofs

We prove security for the case of $k = 2$ servers. All the arguments generalize naturally to the k-server setting with $k > 2$.

Correctness of the scheme introduced in Construction 1 can be verified by inspection. To prove both integrity and security, we find it useful to first prove Lemma 43, which informally states that if a malicious server deviates from the prescribed protocol, the Reconstruct algorithm rejects with high probability.

Lemma 43. Consider the authenticated-PIR scheme in Construction 1, on record size $\ell \in \mathbb{N}$ and with $k = 2$ servers for the sake of the proof. Recall that Construction 1 uses a linear PIR scheme (Definition 42). Then, for every $\lambda \in \mathbb{N}$, every non-zero $\Delta \in \{0,1\}^{\ell_{\mathsf{dig}}+\ell+\ell_\pi}$, where ℓ_{dig} is the length of the digest and ℓ_π is the length of a Merkle inclusion proof as per Definition 35, every database $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$, and every index $i \in [N]$, the following holds:

$$\Pr\left[y \neq \perp : \begin{array}{r} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^\lambda, i) \\ a_1 \leftarrow \mathsf{Answer}(\mathbf{X}, q_1) \\ a_2 \leftarrow \mathsf{Answer}(\mathbf{X}, q_2) \\ y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a_1 \oplus \Delta, a_2) \end{array} \right] \leq \mathsf{negl}(\lambda),$$

where the probability is computed over all the random coins used by the algorithms of the scheme. The statement holds also when the roles of honest and malicious server are

inverted.

Proof. We parse Δ as $(\Delta_{\mathsf{root}}, \Delta_{\mathbf{x}}, \Delta_{\pi})$ where $\Delta_{\mathsf{root}} \in \{0,1\}^{\ell_{\mathsf{dig}}}$, $\Delta_{\mathbf{x}} \in \{0,1\}^{\ell}$, and $\Delta_{\pi} \in \{0,1\}^{\ell_{\pi}}$. If $\Delta_{\mathsf{root}} \neq 0^{\ell_{\mathsf{dig}}}$ then parsing $a_1 + \Delta$ and a_2 yields two different roots and the client immediately rejects (line 3 of Reconstruct in Construction 1). Hence, assume the client gets identical roots from the servers, i.e., $\Delta_{\mathsf{root}} = 0^{\ell_{\mathsf{dig}}}$. The client therefore receives two honest digests of the database $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^{\ell}$.

Assume by contradiction that there is an index $i \in [N]$ and $\Delta = (\Delta_{\mathsf{root}}, \Delta_{\mathbf{x}}, \Delta_{\pi})$ where $\Delta_{\mathsf{root}} = 0^{\ell_{\mathsf{dig}}}$, $\Delta_{\mathbf{x}} \in \{0,1\}^{\ell}$, $\Delta_{\pi} \in \{0,1\}^{\ell_{\pi}}$, and a database $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^{\ell}$ such that

$$\Pr\left[y \neq \bot : \begin{array}{r} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^{\lambda}, i) \\ a_1 \leftarrow \mathsf{Answer}(\mathbf{X}, q_1) \\ a_2 \leftarrow \mathsf{Answer}(\mathbf{X}, q_2) \\ y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a_1 \oplus \Delta, a_2) \end{array}\right] \geq \nu,$$

where ν is *non-negligible* in the security parameter λ. By the assumption that the client gets identical roots from the servers, we can rewrite the above probability as

$$\Pr\left[y \neq \bot : \begin{array}{r} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^{\lambda}, i) \\ (i, \mathsf{st}_{\mathsf{PIR}}) \leftarrow \mathsf{st} \\ a_1 = (\mathsf{root}_1, a_1') \leftarrow \mathsf{Answer}(\mathbf{X}, q_1) \\ a_2 = (\mathsf{root}_2, a_2') \leftarrow \mathsf{Answer}(\mathbf{X}, q_2) \\ r_i \leftarrow \mathsf{PIR.Reconstruct}\left(\mathsf{st}_{\mathsf{PIR}}, a_1' \oplus (\Delta_{\mathbf{x}} \| \Delta_{\pi}), a_2'\right) \\ (\mathbf{x}_i, \pi_i) \leftarrow r_i \\ y \leftarrow \begin{cases} \bot & \mathsf{M.VerifyIncludes}(\mathsf{root}_1, i, \mathbf{x}_i, \pi_i) = \bot \\ \mathbf{x}_i & \text{otherwise} \end{cases} \end{array}\right] \geq \nu.$$

By the linearity of the classic PIR scheme that Construction 1 uses, we can rewrite the above probability as

$$\Pr\left[y \neq \bot : \begin{array}{r} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^{\lambda}, i) \\ a_1 = (\mathsf{root}_1, a_1') \leftarrow \mathsf{Answer}(\mathbf{X}, q_1) \\ a_2 = (\mathsf{root}_2, a_2') \leftarrow \mathsf{Answer}(\mathbf{X}, q_2) \\ r_i \leftarrow a_1' \oplus \Delta_{\mathbf{x}} \| \Delta_{\pi} \oplus a_2' \\ (\mathbf{x}_i, \pi_i) \leftarrow r_i \\ y \leftarrow \begin{cases} \bot & \mathsf{M.VerifyIncludes}(\mathsf{root}_1, i, \mathbf{x}_i, \pi_i) = \bot \\ \mathbf{x}_i & \text{otherwise} \end{cases} \end{array}\right] \geq \nu.$$

We now show that if $\Delta_{\mathbf{x}} \neq 0^{\ell}$, then the malicious servers breaks soundness of the Merkle-

tree scheme (Definition 38). Alternatively, if $\Delta_{\mathbf{x}} = 0^{\ell}$, but $\Delta_{\pi} \neq 0^{\ell_\pi}$, then the malicious server breaks uniqueness of the Merkle-tree scheme (Definition 37).

We analyze the first case, that is, we assume that $\Delta_{\mathbf{x}} \neq 0^{\ell}$. Let $\mathcal{A}$ be an adversary in the definition of soundness for a Merkle-tree scheme (Definition 38). We show how $\mathcal{A}$ can use $\Delta = (\Delta_{\mathsf{root}}, \Delta_{\mathbf{x}}, \Delta_{\pi})$ with $\Delta_{\mathsf{root}} = 0^{\ell_{\mathsf{dig}}}$, $\Delta_{\mathbf{x}} \neq 0^{\ell}$, $\Delta_{\pi} \in \{0,1\}^{\ell_\pi}$ to break the soundness property of the Merkle-tree scheme with a non-negligible probability. Given i, Δ, $\mathbf{X}$, the adversary $\mathcal{A}$ proceeds as follows:

1. Construct a query $(\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^{\lambda}, i)$.
2. For $k \in [2]$, compute $a_k = (\mathsf{root}_k, a'_k) \leftarrow \mathsf{Answer}(\mathbf{X}, q_k)$.
3. Compute $r_i \leftarrow a'_1 \oplus a'_2$ and
4. Parses the reconstructed value as $r_i = (\mathbf{x}_i, \pi_i)$.

Algorithm $\mathcal{A}$ outputs $(\mathbf{X}, i, \mathbf{x}_i \oplus \Delta_{\mathbf{x}}, \pi_i \oplus \Delta_{\pi})$ in the soundness game of Definition 38. By assumption, the digest is correct and computed over $\mathbf{X}$. Since $\Delta_{\mathbf{x}} \neq 0^{\ell}$, we know that

$$(\mathbf{x}_i \oplus \Delta_{\mathbf{x}}) \| (\pi_i \oplus \Delta_{\pi}) \neq \mathbf{x}_i \| \pi_i.$$

Moreover, the probability stated in Definition 38 is equal to the probability stated in this lemma (i.e., to ν). Since by assumption ν is non-negligible in the security parameter λ, algorithm $\mathcal{A}$ successfully breaks the soundness property of the Merkle-tree scheme.

We analyze now the second case, that is, we assume that $\Delta_{\mathbf{x}} = 0^{\ell}$ and $\Delta_{\pi} \neq 0^{\ell_\pi}$. Let $\mathcal{A}'$ be an adversary in the definition of uniqueness for a Merkle-tree scheme (Definition 37). We show how $\mathcal{A}'$ can use $\Delta = (\Delta_{\mathsf{root}}, \Delta_{\mathbf{x}}, \Delta_{\pi})$ with $\Delta_{\mathsf{root}} = 0^{\ell_{\mathsf{dig}}}$, $\Delta_{\mathbf{x}} = 0^{\ell}$, $\Delta_{\pi} \neq 0^{\ell_\pi}$ to break the uniqueness property of the Merkle-tree scheme with a non-negligible probability. Given i, Δ, $\mathbf{X}$, the adversary $\mathcal{A}'$ proceeds exactly as algorithm $\mathcal{A}$:

1. Construct a query $(\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^{\lambda}, i)$.
2. For $k \in [2]$, compute $a_k = (\mathsf{root}_k, a'_k) \leftarrow \mathsf{Answer}(\mathbf{X}, q_k)$.
3. Compute $r_i \leftarrow a'_1 \oplus a'_2$ and
4. Parses the reconstructed value as $r_i = (\mathbf{x}_i, \pi_i)$.

Algorithm $\mathcal{A}'$ outputs $(\mathbf{X}, i, \mathbf{x}_i, \pi_i, \pi_i \oplus \Delta_{\pi})$ in the uniqueness game of Definition 37. Since $\Delta_{\pi} \neq 0^{\ell_\pi}$, we know that $\pi_i \neq \pi_i \oplus \Delta_{\pi}$. Moreover, the probability stated in Definition 37 is equal to the probability stated in this lemma (i.e., to ν). Since by assumption ν is non-negligible in the security parameter λ, $\mathcal{A}'$ successfully breaks the uniqueness property of the Merkle-tree scheme. □

We now use Lemma 43 to show that the scheme presented in Construction 1 ensures integrity and security, and is hence secure.

Theorem 44 (Integrity of Construction 1)**.** The authenticated PIR scheme of Construction 1 provides integrity.

Proof. This follows directly from Lemma 43. □

Theorem 45 (Privacy of Construction 1)**.** The authenticated PIR scheme of Construction 1 provides privacy.

Proof. Recall that Construction 1 is an authenticated PIR scheme for point queries: the Query algorithm inputs an index $i \in [N]$ and $\mathsf{Reconstruct}$ outputs either a vector $\mathbf{x}_i \in \{0,1\}^\ell$ or the rejection symbol $\perp$. Example 25 shows how to recover this functionality from Definition 20.

Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be the adversary of Definition 23. We syntactically change the distribution modeling the real world by introducing an additional variable $\Delta \leftarrow a_1 \oplus a_{\mathcal{A}}$, which gives the following distribution:

$$\mathrm{REAL'}_{\mathcal{A},i,\lambda,\mathbf{X}} = \left\{ \hat{\beta} : \begin{array}{l} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^\lambda, i) \\ a_j \leftarrow \mathsf{Answer}(\mathbf{X}, q_j) \quad \forall j \in [2] \\ \mathsf{st}_{\mathcal{A}}, a_{\mathcal{A}} \leftarrow \mathcal{A}_0(\mathbf{X}, q_1) \\ \mathbf{\Delta} \leftarrow a_1 \oplus a_{\mathcal{A}} \\ y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a_1 \oplus \mathbf{\Delta}, a_2) \\ b \leftarrow \mathbb{1}\{y \neq \perp\} \\ \hat{\beta} \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\},$$

where $\mathbf{\Delta} \in \{0,1\}^{\ell_{\mathsf{dig}}+\ell+\ell_\pi}$, and ℓ_{dig} and ℓ_π are parameters of the Merkle-tree scheme (Definition 35) that Construction 1 uses. Without loss of generality we assume that server 1 is adversarial, i.e., we assign q_1 to the adversary. The proof holds also if we swap the queries.

We additionally adapt the distribution modeling the ideal world to the notation that we use in this proof (by renaming Q to q_1, A to $a_{\mathcal{A}}$ and ignoring weights $\mathbf{w} \in \mathbb{F}^N$ as Construction 1 does):

$$\mathrm{IDEAL'}_{\mathcal{A},\mathcal{S},\mathcal{F},\lambda,\mathbf{X}} = \left\{ \beta : \begin{array}{l} (\mathsf{st}_{\mathcal{S}}, q_1) \leftarrow \mathcal{S}_0(1^\lambda, \mathcal{F}, \mathbf{X}) \\ (\mathsf{st}_{\mathcal{A}}, a_{\mathcal{A}}) \leftarrow \mathcal{A}_0(\mathbf{X}, q_1) \\ b \leftarrow \mathcal{S}_1(\mathsf{st}_{\mathcal{S}}, a_{\mathcal{A}}) \\ \beta \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\}.$$

For any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ let a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ such that for every $\lambda \in \mathbb{N}$, $\mathbf{X} = (\mathbf{x}_i, \ldots, \mathbf{x}_N) \in (\{0,1\}^\ell)^N$ the simulator proceeds as follows, where $\mathcal{S}_{\mathsf{PIR}}$ is the simulator induced by the classic PIR scheme that Construction 1 uses (see Definition 41):

Simulator $\mathcal{S}_0(1^\lambda, \mathcal{F}, \mathbf{X})$	**Simulator** $\mathcal{S}_1(\mathsf{st}_{\mathcal{S}}, a_{\mathcal{A}})$
1 : $q_1 \leftarrow \mathcal{S}_{\mathsf{PIR}}$	1 : $(\mathbf{X}, q_1) \leftarrow \mathsf{st}_{\mathcal{S}}$
2 : $\mathsf{st}_{\mathcal{S}} \leftarrow (\mathbf{X}, q_1)$	2 : $\Delta \leftarrow \mathsf{Answer}(\mathbf{X}, q_1) \oplus a_{\mathcal{A}}$
3 : **return** (st_S, q_1)	3 : $b \leftarrow \mathbb{1}\{\Delta = 0\}$
	4 : **return** b

We now prove that the real and ideal distributions are computationally indistinguishable and hence the scheme presented in Construction 1 provides privacy. To this end, we define five hybrid distributions H_0, H_1, H_2, H_3, H_4:

- H_0: This is the distribution $\mathrm{REAL}'_{\mathcal{A},i,\lambda,\mathbf{X}}$, where we define $\Delta \leftarrow a_1 \oplus a_{\mathcal{A}} = \mathsf{Answer}(\mathbf{X}, q_1) \oplus a_{\mathcal{A}}$ and the bit $b \leftarrow \mathbb{1}\{y \neq \perp\}$ given as input to the adversary $\mathcal{A}_0$ is determined using the output from the $\mathsf{Reconstruct}$ algorithm.

- H_1: Same as H_0 except the hybrid uses $\mathsf{Reconstruct}'$ instead of the $\mathsf{Reconstruct}$ procedure of Construction 1. $\mathsf{Reconstruct}'$ is the same as $\mathsf{Reconstruct}$, except that it computes $r_i \leftarrow a'_1 \oplus a'_2$ instead of $r_i \leftarrow \mathsf{PIR.Reconstruct}(a'_1, a'_2)$ in line 4. The difference between H_0 and H_1 is boxed in the definition below:

$$\mathrm{H}_1 = \left\{ \hat{\beta} : \begin{array}{l} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^\lambda, i) \\ a_j \leftarrow \mathsf{Answer}(\mathbf{X}, q_j) \quad \forall j \in [2] \\ \mathsf{st}_{\mathcal{A}}, a_{\mathcal{A}} \leftarrow \mathcal{A}_0(\mathbf{X}, q_1) \\ \mathbf{\Delta} \leftarrow a_1 \oplus a_{\mathcal{A}} \\ \boxed{y \leftarrow \mathsf{Reconstruct}'(\mathsf{st}, a_1 \oplus \mathbf{\Delta}, a_2)} \\ b \leftarrow \mathbb{1}\{y \neq \perp\} \\ \hat{\beta} \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\},$$

- H_2: Same as H_1 except the hybrid computes the acceptance bit b by checking

whether $\Delta = 0$. The difference between H_1 and H_2 is boxed in the definition below:

$$\mathrm{H}_2 = \left\{ \hat{\beta} : \begin{array}{l} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^\lambda, i) \\ a_j \leftarrow \mathsf{Answer}(\mathbf{X}, q_j) \quad \forall j \in [2] \\ \mathsf{st}_{\mathcal{A}}, a_{\mathcal{A}} \leftarrow \mathcal{A}_0(\mathbf{X}, q_1) \\ \boldsymbol{\Delta} \leftarrow a_1 \oplus a_{\mathcal{A}} \\ y \leftarrow \mathsf{Reconstruct}'(\mathsf{st}, a_1 \oplus \boldsymbol{\Delta}, a_2) \\ \boxed{b \leftarrow \mathbb{1}\{\Delta = 0\}} \\ \hat{\beta} \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\},$$

- H_3: Same as H_3 except the adversary gets a query produced by the simulator $\mathcal{S}_{\mathsf{PIR}}$ induced by the unauthenticated PIR scheme. The difference between H_2 and H_3 is boxed in the definition below:

$$\mathrm{H}_3 = \left\{ \hat{\beta} : \begin{array}{l} (\mathsf{st}, \_, q_2) \leftarrow \mathsf{Query}(1^\lambda, i) \\ \boxed{q_1 \leftarrow \mathcal{S}_{\mathsf{PIR}}} \\ a_j \leftarrow \mathsf{Answer}(\mathbf{X}, q_j) \quad \forall j \in [2] \\ \mathsf{st}_{\mathcal{A}}, a_{\mathcal{A}} \leftarrow \mathcal{A}_0(\mathbf{X}, q_1) \\ \boldsymbol{\Delta} \leftarrow a_1 \oplus a_{\mathcal{A}} \\ y \leftarrow \mathsf{Reconstruct}'(\mathsf{st}, a_1 \oplus \boldsymbol{\Delta}, a_2) \\ b \leftarrow \mathbb{1}\{\Delta = 0\} \\ \hat{\beta} \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\},$$

- H_4: This is the distribution $\mathrm{IDEAL}'_{\mathcal{A},\mathcal{S},\mathcal{F},\lambda,\mathbf{X}}$.

We now argue that each pair of adjacent hybrids is indistinguishable. For $j \in \{0, 1, 2, 3, 4\}$, let W_b the event that the output of the hybrid experiment H_b is "1."

- Hybrid H_1 is the same as hybrid H_0 except H_1 uses $\mathsf{Reconstruct}'$ instead of $\mathsf{Reconstruct}$. As Construction 1 uses a linear classic PIR scheme, these hybrids are equal, i.e.,
$$|\Pr[W_0] - \Pr[W_1]| = 0.$$
- Hybrid H_2 is the same as hybrid H_1 except H_2 computes the acceptance but b by checking whether $\Delta = 0$. If $\boldsymbol{\Delta} = 0^{\ell_{\mathsf{dig}}+\ell+\ell_\pi}$ (i.e, a binary string of $\ell_{\mathsf{dig}} + \ell + \ell_\pi$ zeros) the simulator $\mathcal{S}_1$ sets $b = 1$; if $\boldsymbol{\Delta} \neq 0^{\ell_{\mathsf{dig}}+\ell+\ell_\pi}$, then $\mathcal{S}_1$ sets $b = 0$. By Lemma 43 we know that
$$|\Pr[b \leftarrow \mathbb{1}\{y \neq \perp\}] - \Pr[b \leftarrow \mathbb{1}\{\Delta = 0\}]| \leq \mathsf{negl}(\lambda),$$
where the first probability refers to the assignment of bit b in H_1, while the second refers to the assignment of bit b in H_2. As the only difference between the two

hybrids is how they set bit b, we can rewrite the above probability as

$$|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{negl}(\lambda).$$

- The only difference between hybrids H_2 and H_3 is how the query q_1 is sampled, i.e., how the query that the adversary gets is sampled. By security of the classic unauthenticated PIR scheme (Definition 41), we have

$$|\Pr[W_2] - \Pr[W_3]| \leq \mathsf{negl}(\lambda).$$

- H_4 is a rewriting of H_3. In H_3, $\mathcal{A}_0$ inputs $q_1 \leftarrow \mathcal{S}_{\mathsf{PIR}}$, where $\mathcal{S}_{\mathsf{PIR}}$ if the simulator induced by the classic PIR scheme; the same happens in H_4. In H_3, $\mathcal{A}_1$ inputs $b \leftarrow \mathbb{1}\{\Delta = 0\}$ and the same happens in H_4. We show the bridging from H_3 to H_4 below, where we rewrite $\Delta \leftarrow a_1 \oplus a_{\mathcal{A}}$ as $\Delta \leftarrow \mathsf{Answer}(\mathbf{X}, q_1)$ and grey lines can be removed from H'_3 to get H_4:

$$\mathrm{H}'_3 = \left\{ \hat{\beta} : \begin{array}{l} (\mathsf{st}, \_, q_2) \leftarrow \mathsf{Query}(1^\lambda, i) \\ q_1 \leftarrow \mathcal{S}_{\mathsf{PIR}} \\ a_j \leftarrow \mathsf{Answer}(\mathbf{X}, q_j) \quad \forall j \in [2] \\ \mathsf{st}_{\mathcal{A}}, a_{\mathcal{A}} \leftarrow \mathcal{A}_0(\mathbf{X}, q_1) \\ \mathbf{\Delta} \leftarrow \mathsf{Answer}(\mathbf{X}, q_1) \oplus a_{\mathcal{A}} \\ y \leftarrow \mathsf{Reconstruct}'(\mathsf{st}, a_1 \oplus \mathbf{\Delta}, a_2) \\ b \leftarrow \mathbb{1}\{\Delta = 0\} \\ \hat{\beta} \leftarrow \mathcal{A}_1(\mathsf{st}_{\mathcal{A}}, b) \end{array} \right\}.$$

Since the hybrids are equal we have

$$|\Pr[W_3] - \Pr[W_4]| = 0.$$

By a standard hybrid argument we conclude that $\mathrm{REAL'}_{\mathcal{A},i,\lambda,\mathbf{X}} \approx_c \mathrm{IDEAL'}_{\mathcal{A},\mathcal{S},\mathcal{F},\lambda,\mathbf{X}}$ and therefore

$$\mathrm{REAL}_{\mathcal{A},i,\lambda,\mathbf{X}} \approx_c \mathrm{IDEAL}_{\mathcal{A},\mathcal{S},\mathcal{F},\lambda,\mathbf{X}}. \qquad \square$$

3.4.2 Predicate queries via function sharing

Recent work on function secret sharing [BGI15, BGI16] in the multi-server PIR setting enables a client to compute a non-trivial function f over the database contents, without revealing this function f to the servers. For example, a client can count the number of database records that match a certain predicate, without revealing this predicate to the servers.

We design an authenticated-PIR protocol for predicate queries by extending classic PIR

schemes based on function secret sharing [BGI15, BGI16]. At a high level, the client makes two correlated PIR queries. The reconstructed answer to the first query should contain the value v that the client wants. The reconstructed answer to the second query should contain $v' = \alpha v$, where α is a random scalar known only to the client. To authenticate the servers' answers, the client checks that $\alpha v = v'$ and rejects if not. As we will show, if any server misbehaves, the client will be checking that $\alpha(v + \Delta) = v' + \Delta'$, for some non-zero Δ and Δ'. Sampling α from a sufficiently large space of values ensures that the client catches a cheating server almost certainly.

This idea of using secret-shared random values for data authentication follows a long line of work on information-theoretic message authentication codes and malicious-secure multiparty computation [DPSZ12, CDF+08, BCG+21, dCP22].

We begin by presenting some preliminaries, followed by a detailed description of our construction.

Preliminary: function secret sharing

We recall the definition of function secret sharing [BGI15, BGI16]. A k-party *function secret-sharing* scheme is defined with respect to a function class $\mathcal{F}$. Each function $f \in \mathcal{F}$ maps elements in some input space to a finite group or field $\mathbb{F}$.

Definition 46 (Function secret sharing)**.** A k party *function secret sharing* scheme for a function class $\mathcal{F}$ defined over a field $\mathbb{F}$ consists of the following two efficient algorithms:

- $\mathsf{Gen}(1^\lambda, f) \to (f_1, \ldots, f_k)$. Given a function $f \in \mathcal{F}$, output k function-secret-shares $f_1, \ldots, f_k$.
- $\mathsf{Eval}(f_i, x) \to f_i(x) \in \mathbb{F}$. Given a secret-share f_i and a function input x, output the evaluation of f_i on x.

A function secret-sharing (FSS) scheme must satisfy the following properties, which we state formally and informally.

Correctness. Given shares $(f_1, \ldots, f_k)$ of a function $f \in \mathcal{F}$, for all x in the domain of f, it holds that $\sum_{i \in [k]} \mathsf{Eval}(f_i, x) = f(x) \in \mathbb{F}$.

Definition 47 (FSS correctness)**.** A k-party function secret-sharing scheme $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ for a function class $\mathcal{F}$ defined over a field $\mathbb{F}$ satisfies *correctness* if for every x in the domain of f, the following holds:

$$\Pr\left[\sum_{i \in [k]} \mathsf{Eval}(f_i, x) = f(x) \in \mathbb{F} : (f_1, \ldots, f_k) \leftarrow \mathsf{Gen}(1^\lambda, f)\right] = 1.$$

Security. Given shares $(f_1, \ldots, f_k)$ of a function $f \in \mathcal{F}$, a computationally-bounded adversary that learns $k-1$ of the shares learns nothing about the shared function f, beyond the fact that $f \in \mathcal{F}$.

Definition 48 (FSS security)**.** Let $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ be a k-party function secret-sharing scheme for a function class $\mathcal{F}$. Let S be any subset of $k-1$ elements from $[k]$. For a security parameter $\lambda \in \mathbb{N}$ and a function $f \in \mathcal{F}$ let the distribution

$$\mathrm{REAL}_{\lambda,f} = \left\{ \bigcup_{i \in S} f_i : (f_1, \ldots, f_k) \leftarrow \mathsf{Gen}(1^\lambda, f) \right\}.$$

Similarly, for a simulator $\mathcal{S}$ let the distribution

$$\mathrm{IDEAL}_{\mathcal{S},\lambda,\mathcal{F}} = \left\{ \{f_i\}_{i \in S} \leftarrow \mathcal{S}(1^\lambda, \mathcal{F}) \right\}$$

A k-party function secret-sharing scheme $\mathsf{FSS} = (\mathsf{Gen}, \mathsf{Eval})$ for a function class $\mathcal{F}$ is *secure* if there exists a simulator $\mathcal{S}$ such that for every security parameter $\lambda \in \mathbb{N}$ and every function $f \in \mathcal{F}$, the following holds:

$$\mathrm{REAL}_{\lambda,f} \approx_c \mathrm{IDEAL}_{\mathcal{S},\lambda,\mathcal{F}}.$$

For the construction, we need the following additional definition:

Definition 49 (Function class closed under scalar multiplication)**.** Let $\mathcal{F}$ be a class of functions whose codomain is a finite field $\mathbb{F}$. Then we say that the function class $\mathcal{F}$ is *closed under scalar multiplication* if, for all functions $f \in \mathcal{F}$ and for all scalars $\alpha \in \mathbb{F}$, it holds that the function $\alpha \cdot f \in \mathbb{F}$.

Construction

Our scheme, presented in Construction 2, is defined with respect to a finite field $\mathbb{F}$, a record length $\ell \in \mathbb{N}$, a database size $N \in \mathbb{N}$, a function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^\ell, \mathbb{F}]$ closed under scalar multiplication, and weights $w \in \mathbb{F}^N$. The $k \geq 2$ servers each hold a copy of a database of N ℓ-bit records. We write the n database records as $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$. Given a predicate function $f \in \mathcal{F}$, the client samples a random non-zero field element $\alpha \in \mathbb{F}$ and secret-shares f together with a new function g defined as $g(i, x_i) = \alpha \cdot f(i, x_i) \in \mathbb{F}$ into k shares, i.e., f_j and g_j for $j \in [k]$. (Alternatively, if the underlying function-secret-sharing scheme supports it, the client can also secret share the single function $(f(i, x_i), g(i, x_i))$ whose image is in $\mathbb{F}^2$.)

Upon receiving the shares, each server $j \in [k]$ sets each element of its answer tuple to the sum of the function shares' evaluations on all the database records multiplied by

Construction 2 (k-server authenticated PIR for predicate queries tolerating $k-1$ malicious servers)**.** The construction is parametrized by a number of servers $k \in \mathbb{N}$, a number of database rows $N \in \mathbb{N}$, a row length $\ell \in \mathbb{N}$, a finite field $\mathbb{F}$, a security parameter λ, a function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^\ell, \mathbb{F}]$ that is closed under scalar multiplication, and a function-secret-sharing scheme $(\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$ for the function class $\mathcal{F}$, parametrized by λ. We represent the database as N binary strings, each of length ℓ: $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$.

$\mathsf{Query}\left(1^\lambda, f\right) \to (\mathsf{st}, \mathbf{q}_1, \ldots, \mathbf{q}_k)$

1. Sample a random field element $\alpha \xleftarrow{\mathrm{R}} \mathbb{F} \setminus \{0\}$.
2. Set the state $\mathsf{st} \leftarrow \alpha$.
3. Let $g \leftarrow \alpha \cdot f$. Such a g must exist since the function class $\mathcal{F}$ is closed under scalar multiplication, as in Definition 49.
4. Compute $q_1, \ldots, q_k \leftarrow \mathsf{FSS.Gen}(1^\lambda, f)$ together with $q'_1, \ldots, q'_k \leftarrow \mathsf{FSS.Gen}(1^\lambda, g)$.
5. Output $(\mathsf{st}, (q_1, q'_1), \ldots, (q_k, q'_k))$.

$\mathsf{Answer}\left(\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell, \mathbf{w} \in \mathbb{F}^N, \mathbf{q}\right) \to \mathbf{a} \in \mathbb{F}^2$

1. Parse $\mathbf{q}$ as (q_f, q_g).
2. Compute answer as $a_f \leftarrow \sum_{j \in [N]} w_j \cdot \mathsf{FSS.Eval}(q_f, \mathbf{x}_j)$ and $a_g \leftarrow \sum_{j \in [N]} w_j \cdot \mathsf{FSS.Eval}(q_g, \mathbf{x}_j)$.
3. Return $\mathbf{a} \leftarrow (a_f, a_g) \in \mathbb{F}^2$.

$\mathsf{Reconstruct}\left(\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{F}^2\right) \to \mathbb{F} \cup \{\perp\}$

1. Parse the state st as $\alpha \in \mathbb{F}$.
2. Compute $\mathbf{a} \leftarrow \mathbf{a}_1 + \cdots + \mathbf{a}_k \in \mathbb{F}^2$.
3. Parse $\mathbf{a}$ as $(m, \tau) \in \mathbb{F}^2$.
4. Compute $\tau' \leftarrow m \cdot \alpha \in \mathbb{F}$.
5. If $\tau = \tau'$, output $m \in \mathbb{F}$. Otherwise, output $\perp$.

the corresponding weights: i.e., $\mathbf{a}_j \leftarrow \left(\sum_{i\in[N]} w_i \cdot f(i, \mathbf{x}_i), \sum_{i\in[N]} w_i \cdot g(i, \mathbf{x}_i)\right) \in \mathbb{F}^2$. The servers directly evaluate the function shares on the database records. The client adds the answer vectors and reconstructs an intermediate value $\mathbf{a} \leftarrow \sum_{j\in[k]} \mathbf{a}_j \in \mathbb{F}^2$.

If all the servers are honest, the client-reconstructed value $\mathbf{a}$ equals $\mathbf{a} = (a_1, a_2) = \left(\sum_{i\in[N]} w_i \cdot f(i, \mathbf{x}_i), \alpha \cdot \sum_{i\in[N]} w_i \cdot f(i, \mathbf{x}_i)\right)$. The client then verifies that $\alpha \cdot a_1 = a_2$. As α is randomly generated and secret-shared among the servers, only the client knows its value. If $\alpha \cdot a_1 \neq a_2$, then the client rejects. Otherwise, the client accepts and outputs a_1.

Proof sketch. To explain how this approach protects integrity, we argue by contradiction. Say that server $j \in [k]$ should have returned an answer $\mathbf{a}_j \in \mathbb{F}^2$ to the client. Suppose server j is malicious and returns an answer $\hat{\mathbf{a}}_j = \mathbf{a}_j + \boldsymbol{\Delta} \in \mathbb{F}^2$ for some non-zero value $\boldsymbol{\Delta} = (\Delta_m, \Delta_\tau) \in \mathbb{F}^2$. The client will reconstruct the answer as $\mathbf{a} + \Delta = \left(\sum_{i\in[N]} w_i \cdot f(i, \mathbf{x}_i) + \Delta_m, \alpha \cdot \sum_{i\in[N]} w_i \cdot f(i, \mathbf{x}_i) + \Delta_\tau\right) \in \mathbb{F}^2$. As server j has no information about α—due to the privacy guarantees of the function-secret-sharing scheme—the malicious server's choice of Δ is (computationally) independent of α. For the verification to pass, it must be that $\alpha \cdot \Delta_m = \Delta_\tau$. If $\boldsymbol{\Delta} \neq 0$ and α is sampled independent of $\boldsymbol{\Delta}$, this happens with probability at most $1/(|\mathbb{F}| - 1)$ over the randomness of α. Next, the privacy of the client's queries is ensured by the underlying function secret-sharing scheme. In Appendix B.2.1, we formally prove that this construction is secure.

Theorem 50. Suppose there exists a k-party function-secret-sharing scheme for a function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^\ell, \mathbb{F}]$ that is closed under scalar multiplication (Definition 49), for database size $N \in \mathbb{N}$, which, on security parameter $\lambda \in \mathbb{N}$, outputs secret shares of length $L(\lambda)$. Then, there is a k-server authenticated-PIR scheme for function class $\mathcal{F}$ with query complexity $2L(\lambda)k$ bits and answer complexity $2k\lambda$ bits.

By applying the two-party function-secret-sharing scheme proposed by Boyle, Gilboa, and Ishai [BGI16], we get:

Corollary 51. Given a length-doubling pseudorandom generator with seed length λ, there is a two-server authenticated PIR scheme for point functions and interval functions with communication complexity $O(\lambda \log N)$, on security parameter λ and database size N.

Handling functions with larger output. In some PIR applications, a client might want to evaluate a function whose output is larger than a single field element, e.g., geographical coordinates for route planners [WYG+17]. We hence extend our scheme to support multi-element authenticated output.

Here, we authenticate each output element of a function f with a separate function g_j, for $j \in [b]$, where b is the output length of f using an algebraic manipulation detection code [CDF+08]. In the query algorithm, the client generates a secret random scalar α as before but then computes $(g_1(i, \mathbf{x}_i), g_2(i, \mathbf{x}_i), \ldots, g_b(i, \mathbf{x}_i)) = (\alpha, \alpha^2, \ldots, \alpha^b) \odot f(i, \mathbf{x}_i)$,

where $\odot$ represents the element-wise product, and sends secret-shared f and $g_1, \ldots, g_b$ to the servers. The servers then compute their answer as $\mathbf{a} \leftarrow (\mathbf{a}_f, \mathbf{a}_{g_1}, \ldots, \mathbf{a}_{g_b}) \in \mathbb{F}^{2b}$.

This already enables the client to validate integrity of the full output after the reconstruction by comparing it with $\mathbf{a}_{g_1}, \ldots, \mathbf{a}_{g_b}$. We further reduce the protocol's communication cost by setting the servers' answer to $(\mathbf{a}_f, \mathbf{a}_g = \sum_{i \in [b]} \mathbf{a}_{g_i}) \in \mathbb{F}^{b+1}$. The client re-computes this linear combination from the answer and compares it with the received value.

We show the full construction in Appendix B.2.2.

3.5 Single-server authenticated PIR

We now present two single-server authenticated-PIR schemes.

As depicted in Figure 3.1, in this setting a data owner outsources the data to a single PIR server (e.g., an Amazon EC2 instance) and produces a database digest. This public digest serves as a commitment to the database contents. The client can fetch the digest from a distributed authority, or using a CONIKS-like gossip protocol [MBB+15], or out-of-band from the data owner.

It is possible in principle to construct single-server authenticated-PIR schemes by augmenting a standard single-server PIR scheme [HHCG+23, DPC23, MCR21, ACLS18, MW22] with a succinct proof of correct server execution [PHGR13], but this would be orders of magnitude more costly in computation than our schemes are.

Preliminary: rebalancing to get $\sqrt{N}$ communication

The single-server authenticated-PIR schemes natively have a digest of size $\mathsf{poly}(\lambda)$ bits, upload $N \cdot \mathsf{poly}(\lambda)$ bits, and download $\mathsf{poly}(\lambda)$ bits. To reduce total communication to $\sqrt{N} \cdot \mathsf{poly}(\lambda)$ bits, we use a standard rebalancing trick [CGKS95]. We summarize that technique before detailing our schemes.

The server first splits the database into $\sqrt{N}$ chunks, each of size $\sqrt{N}$. The digest then consists of the hash (with any collision-resistant hash function, e.g., SHA-256) of the $\sqrt{N}$ database digests. To query the database for the i^{th} row of the j^{th} chunk, the client issues a single query for row i. The server responds with the $\sqrt{N}$ chunk digests, and the answer computed against each chunk. The client checks that (1) the hash of the $\sqrt{N}$ chunk digests match the database digest and (2) all $\sqrt{N}$ chunk queries accept. If these checks pass, the client outputs the value of the j^{th} response as its answer.

3.5.1 From learning with errors

Our first single-server authenticated-PIR scheme builds on lattices and relies on the learning-with-errors assumption (LWE) [Reg05].

Preliminaries: lattices and the learning-with-errors assumption

The LWE assumption with parameters $n, q, m, s \in \mathbb{N}$, states that the two distributions $(\mathbf{A}, \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal)$ and $(\mathbf{A}, \mathbf{u}^\intercal)$ are computationally indistinguishable, where $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},s}^m \in \mathbb{Z}_q^m$, and $\mathbf{u} \xleftarrow{R} \mathbb{Z}_q^n$, and where $D_{\mathbb{Z},s}$ is the discrete-Gaussian distribution with width parameter s. We introduce these concepts formally in what follows.

For a real value $s > 0$, we write $\rho_s \colon \mathbb{R} \to \mathbb{R}^+$ to denote the Gaussian function $\rho_s(x) := \exp(-\pi x^2/\sigma^2)$. The discrete Gaussian distribution $D_{\mathbb{Z},s}$ with width parameter s is a discrete distribution over the integers with probability mass function

$$\Pr[X = x : X \leftarrow D_{\mathbb{Z},s}] = \frac{\rho_s(x)}{\sum_{y\in\mathbb{Z}} \rho_s(y)}.$$

We say that a distribution D (over $\mathbb{R}$) is subgaussian with parameter s if for every $t \geq 0$,

$$\Pr[|x| > t : x \leftarrow D] \leq 2\exp(-\pi t^2/s^2). \tag{3.1}$$

The discrete Gaussian distribution $D_{\mathbb{Z},s}$ is *subgaussian* with parameter s. In particular, this means that if we sample $e \leftarrow D_{\mathbb{Z},s}$, then $|e| \leq \sqrt{\lambda}s$ with probability $1 - \mathsf{negl}(\lambda)$. Moreover, if x_1, x_2 are independent subgaussian random variables with parameters s_1, s_2, then $x = \alpha x_1 + \beta x_2$ is subgaussian with parameter $\sqrt{\alpha^2 s_1^2 + \beta^2 s_2^2}$ for any $\alpha, \beta \in \mathbb{R}$.

In the following description, unless otherwise noted, all operations are performed over $\mathbb{Z}_q$. For a value $x \in \mathbb{Z}_q$, we write $|x|$ to denote the absolute value of its canonical representative in the interval $\mathbb{Z} \cap [-q/2, q/2]$.

We now recall the learning with errors assumption [Reg05]:

Definition 52 (Learning with Errors [Reg05])**.** Let λ be a security parameter. Let $n = n(\lambda)$ be the lattice dimension, $m = m(\lambda)$ be the number of samples, $q = q(\lambda)$ be a modulus, and $s = s(\lambda)$ be a Gaussian width parameter. Then, the learning with errors (LWE) assumption $\mathsf{LWE}_{n,m,q,s}$ states that the following distributions are computationally indistinguishable:

$$(\mathbf{A}, \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal) \approx_c (\mathbf{A}, \mathbf{u}^\intercal),$$

where $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},s}^m$, and $\mathbf{u} \xleftarrow{R} \mathbb{Z}_q^n$.

The security of our construction will rely on the "extended LWE" assumption [BLP+13],

which essentially says that LWE holds even if the distinguisher learns a linear combination of the LWE errors. We state the assumption below:

Definition 53 (Extended LWE [BLP+13])**.** Let λ be a security parameter and let $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$, and $s = s(\lambda)$ be lattice parameters (as in Definition 52). Then, the extended learning with errors (extLWE) assumption $\mathsf{extLWE}_{n,m,q,s}$ states that for every $\mathbf{x} \in \{0,1\}^m$, the following distributions are computationally indistinguishable:

$$(\mathbf{A}, \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal, \mathbf{e}^\intercal \mathbf{x}) \approx_c (\mathbf{A}, \mathbf{u}^\intercal, \mathbf{e}^\intercal \mathbf{x}),$$

where $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},s}^m$, and $\mathbf{u} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$. More precisely, for an adversary $\mathcal{A}$, we write $\mathsf{Adv}_{\mathsf{extLWE}}^{(n,m,q,s)}[\mathcal{A}]$ to denote the distinguishing advantage of $\mathcal{A}$ for the aforementioned distributions.

Previously, Brakerski et al. [BLP+13, Lemmas 4.3, 4.7] showed that hardness of the extended LWE assumption $\mathsf{extLWE}_{n,m,q,s}$ can be based on the hardness of the vanilla LWE assumption $\mathsf{LWE}_{n,m,q,s'}$ for $s' = O(s)$.

Construction

Construction 3 describes our scheme, which is a twist on Regev's LWE-based encryption scheme [Reg05] and is an authenticated analogue of the SimplePIR LWE-based PIR scheme [HHCG+23]. (We compare against SimplePIR in Section 3.7.) Regev's scheme encrypts a vector $\mathbf{v} \in \{0,1\}^N \subseteq \mathbb{Z}_q^N$ by the pair $(\mathbf{A}, \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal + t \cdot \mathbf{v}^\intercal)$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times N}$ is the LWE matrix, $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ is the LWE secret, $\mathbf{e} \leftarrow D_{\mathbb{Z},s}^N$ is the error vector, and $t \in \mathbb{Z}_q$ is some scaling factor (commonly set to $q/2$). Regev's scheme is linearly homomorphic: for any vector $\mathbf{x} \in \{0,1\}^N \subseteq \mathbb{Z}_q^N$, the ciphertext $(\mathbf{A}\mathbf{x}, (\mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal + t \cdot \mathbf{v}^\intercal) \cdot \mathbf{x})$ decrypts to $\mathbf{v}^\intercal \mathbf{x}$ (provided the accumulated error $\mathbf{e}^\intercal \mathbf{x}$ is small compared to t).

In our scheme, the first portion of this ciphertext ($\mathbf{A} \cdot \mathbf{x}$, on database $\mathbf{x} \in \{0,1\}^N \subseteq \mathbb{Z}_q^N$) becomes the digest. Finding two distinct databases that map to the same digest is as hard as solving the short integer solutions problem [Ajt96].

To query for database record $i \in [N]$, the client prepares the Regev encryption $\mathbf{q}^\intercal$ of the i^{th} basis vector $\boldsymbol{\eta}_i \in \mathbb{Z}_q^N$ (i.e., $\boldsymbol{\eta}_i$ is the vector that is 0 everywhere and 1 at index i). The scaling factor $t \in \mathbb{Z}_q$ is sampled randomly (from an appropriate range), which is critical for the security analysis. To answer the query, the server homomorphically computes the encryption of the inner product of the client's query with the database: $\mathbf{q}^\intercal \mathbf{x} \in \mathbb{Z}_q$. The client checks that the decrypted value is either 0 (indicating a database bit of zero) or close to t (indicating a database bit of one). Otherwise, the client outputs $\perp$.

Finally, by rebalancing Construction 3, we have:

Theorem 54. Under the LWE assumption, Construction 3 is a secure single-server

authenticated-PIR scheme when instantiated with database size N, lattice parameters (n, q, s), random matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times N}$, and bound $B = O(\sqrt{\lambda N} s)$. The digest size consists of $n\sqrt{N}$ elements of $\mathbb{Z}_q$ and the per-query communication cost is $2\sqrt{N}$ elements of $\mathbb{Z}_q$. The scheme has integrity error $\epsilon < 2B/(q - 4B)$.

Proof. The theorem follows from the three theorems that we prove in Appendix B.3. Theorem 71 establishes that Construction 3 is correct. Theorem 73 states that Construction 3 has integrity error $\epsilon < 2B/(q - 4B)$ for bound $B = O(\sqrt{\lambda N} s)$ and lattice parameters (n, q, s). Theorem 74 demonstrates that Construction 3 provides privacy. Therefore, Construction 3 is secure. □

The most important difference between SimplePIR [HHCG+23] and Construction 3 is in the choice of LWE parameters. Since the integrity error is roughly $\sqrt{N}/q$, on database size N and modulus q, we must take the modulus q to be at least 128 bits to achieve negligible integrity error. (Alternatively, we can use a smaller modulus and run the protocol many times to amplify integrity as per Section 3.3.3.) In contrast, SimplePIR uses a 32-bit modulus with no repetition.

3.5.2 From decisional Diffie-Hellman

This second construction uses the decisional Diffie-Hellman assumption.

Preliminary: the decisional Diffie-Hellman assumption (DDH)

The DDH assumption holds in a group $\mathbb{G}$ of prime order p generated by $g \in \mathbb{G}$, if for $x, y, z \xleftarrow{R} \mathbb{Z}_p$, the two distributions (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z) are computationally indistinguishable.

Definition 55 (Decisional Diffie-Hellman)**.** Let λ be a security parameter and let $\mathbb{G}$ be a group of prime order p where $1/p = \mathsf{negl}(\lambda)$. Let g be a generator of $\mathbb{G}$. We say that the decisional Diffie-Hellman assumption (DDH) holds in $\mathbb{G}$ if the following distributions are computationally indistinguishable:

$$(g, h, g^x, h^x) \approx_c (g, h, g^x, z),$$

where $h, z \xleftarrow{R} \mathbb{G}$ and $x \xleftarrow{R} \mathbb{Z}_p$.

By a random self-reduction [Sta96, NR97], one can show that if the DDH assumption holds in $\mathbb{G}$, then for all polynomials $N = N(\lambda)$, the following distributions are also

Construction 3 (Single-server authenticated PIR from LWE)**.** The construction is parametrized by a database length $N \in \mathbb{N}$, a lattice dimension $n \in \mathbb{N}$, a modulus $q \in \mathbb{N}$, a Gaussian width parameter $s \in \mathbb{N}$, a bound $B \in \mathbb{N}$, and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times N}$. The database is a vector $\mathbf{x} \in \{0,1\}^N$.

$\underline{\mathsf{Digest}(\mathbf{x} \in \{0,1\}^N) \to \mathbf{d} \in \mathbb{Z}_q^n}$

1. Output $\mathbf{d} \leftarrow \mathbf{A}\mathbf{x} \in \mathbb{Z}_q^n$.

$\underline{\mathsf{Query}\left(\mathbf{d} \in \mathbb{Z}_q^N, i \in [N]\right) \to (\mathsf{st}, \mathbf{q})}$

1. Sample $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},s}^N \in \mathbb{Z}_q^m$, and $t \xleftarrow{\text{R}} \left[2B, q - 2B\right]$. (Here $D_{\mathbb{Z},s}$ denotes the discrete Gaussian distribution over $\mathbb{Z}$ with parameter s.)
2. Compute $\mathbf{q}^\mathsf{T} \leftarrow \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T} + t \cdot \boldsymbol{\eta}_i^\mathsf{T} \in \mathbb{Z}_q^m$, where $\boldsymbol{\eta}_i \in \mathbb{Z}_q^N$ denotes the i^{th} standard basis vector (i.e., the vector that is 0 everywhere except 1 in index i).
3. Set $\mathsf{st} \leftarrow (\mathbf{d}, \mathbf{s}, t)$ and output $(\mathsf{st}, \mathbf{q})$.

$\underline{\mathsf{Answer}\left(\mathbf{d} \in \mathbb{Z}_q^n, \mathbf{x} \in \{0,1\}^N \subseteq \mathbb{Z}_q^N, \mathbf{q} \in \mathbb{Z}_q^N\right) \to a \in \mathbb{Z}_q}$

1. Output $a \leftarrow \mathbf{q}^\mathsf{T}\mathbf{x} \in \mathbb{Z}_q$

$\underline{\mathsf{Reconstruct}\,(\mathsf{st}, a) \to \{0, 1, \perp\}}$

1. Parse the state st as $(\mathbf{d}, \mathbf{s}, t)$.
2. If there exists $k \in \{0,1\}$ such that $|a - \mathbf{s}^\mathsf{T}\mathbf{d} - kt| < B$, then output k. Otherwise, output $\perp$.

computationally indistinguishable:

$$(g, h_1, h_1^r, \ldots, h_N, h_N^r) \approx_c (g, h_1, z_1, \ldots, h_N, z_N), \tag{3.2}$$

where $h_1, \ldots, h_N, z_1, \ldots, z_N \xleftarrow{\text{R}} \mathbb{G}$ and $r \xleftarrow{\text{R}} \mathbb{Z}_p$.

Construction

Construction 4 details our scheme, which uses a group $\mathbb{G}$ of large prime order p. The database is a vector of N bits $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$. The public parameters of the scheme include group elements $h_1, \ldots, h_N \in \mathbb{G}$. The digest is the product $d \leftarrow \prod_{j=1}^N h_j^{x_j} \in \mathbb{G}$. Finding two distinct databases that map to the same digest is as hard as solving the discrete-log problem in $\mathbb{G}$ [Ped91].

The protocol operates as follows. The client samples two random values $r, t \xleftarrow{\text{R}} \mathbb{Z}_p$. The client then prepares a vector of N group elements. Say the client wants to fetch the i^{th}

Construction 4 (Single-server authenticated PIR from DDH)**.** The construction is parametrized by a database length $N \in \mathbb{N}$, a group $\mathbb{G}$ of prime order p, and group elements $h_1, \ldots, h_N \in \mathbb{G}$. The database is a vector $\mathbf{x} \in \{0,1\}^\ell \subseteq \mathbb{Z}_p^N$.

$\underline{\mathsf{Digest}(\mathbf{x} \in \{0,1\}^N) \to d \in \mathbb{G}}$

1. Output $d \leftarrow \prod_{j \in [N]} h_j^{x_j} \in \mathbb{G}$.

$\underline{\mathsf{Query}\,(d \in \mathbb{G}, i \in [N]) \to (\mathsf{st}, q)}$

1. Sample two random values $r, t \xleftarrow{\text{R}} \mathbb{Z}_p$.
2. For $j \in [N] \backslash \{i\}$, compute $q_j \leftarrow h_j^r \in \mathbb{G}$.
3. Compute $q_i \leftarrow h_i^{r+t} \in \mathbb{G}$.
4. Set $\mathsf{st} \leftarrow (i, d, r, t)$.
5. Set $\mathbf{q} \leftarrow (q_1, \ldots, q_N) \in \mathbb{G}^N$.
6. Output $(\mathsf{st}, \mathbf{q})$.

$\underline{\mathsf{Answer}\left(d \in \mathbb{G}, \mathbf{x} \in \{0,1\}^N \subseteq \mathbb{Z}_p^N, \mathbf{q}\right) \to a \in \mathbb{G}}$

1. Parse the query $\mathbf{q}$ as $(q_1, \ldots, q_N) \in \mathbb{G}^N$.
2. Output $a \leftarrow \prod_{j \in [N]} q_j^{x_j} \in \mathbb{G}$.

$\underline{\mathsf{Reconstruct}\,(\mathsf{st}, a) \to \{0, 1, \perp\}}$

1. Parse the state st as (i, d, r, t).
2. Set $m \leftarrow d^{-r} \cdot a \in \mathbb{G}$.
3. If $m = 1_\mathbb{G}$, output "0." If $m = h_i^t$, output "1."
   Otherwise, output $\perp$.

database bit. For $j \in [N]$, the j^{th} component of this vector is $q_j \leftarrow h_j^{r+t}$ if $j = i$ and is $q_j \leftarrow h_j^r$ otherwise. Under DDH, the server cannot differentiate between q_i and q_j for $j \neq i$.

The client queries the server with the resulting blinded vector $(q_1, \ldots, q_N)$. The server exponentiates each vector element to the corresponding database bit and computes the product $a = \prod_{j \in [N]} q_j^{x_j}$. If the server honestly executes the protocol, the client receives back the product of the blinded digest d^r and (a) either the group identity (when the retrieved bit is zero) or (b) the blinding factor h^t associated with the element of interest (when the retrieved bit is one). If the server returns any answer apart from the one prescribed by the protocol, the client detects this and rejects with overwhelming probability.

We then have, by rebalancing Construction 4:

Theorem 56. If the DDH assumption holds in group $\mathbb{G}$, then Construction 4 is a secure single-server authenticated-PIR scheme when instantiated with database size N and group $\mathbb{G}$. The digest size consists in $\sqrt{N}$ elements of $\mathbb{G}$ and the per-query communication cost is $2\sqrt{N}$ elements of $\mathbb{G}$. The scheme has negligible integrity error.

Proof. Correctness can be verified by inspection. Then the theorem follows from two theorems that we prove in Appendix B.4.1. Theorem 76 states that Construction 4 has a negligible integrity error. Theorem 77 demonstrates that Construction 4 provides privacy. Therefore, Construction 4 is secure. □

The scheme could be extended to retrieve multi-bit database entries in two readily-apparent ways. The first and simplest approach is to run Construction 4 in parallel for each bit of the entry. The second approach requires the client to solve tractable discrete logarithms, as we describe in Appendix B.4.2.

Incremental digest maintenance. We envision that the data owner would generate the database digest and publish it on a client-accessible website or a tamper-resistant log. If a database record changes, the data owner can update the digest in either construction incrementally. For example, in the lattice based construction given an old digest $\mathbf{d} = \mathbf{A}\mathbf{x}$ and a new database $\mathbf{x}'$, the new digest is $\mathbf{d}' = \mathbf{d} + \mathbf{A}(\mathbf{x}' - \mathbf{x})$. Given the old digest, the server can compute the new digest in time proportional to the cost of computing $\mathbf{A}(\mathbf{x}' - \mathbf{x})$. This matrix-vector product, in turn, takes time linear in the number of updates to the database, i.e., the Hamming weight of the difference $\mathbf{x}' - \mathbf{x}$. If the database itself is public, any third party can verify that the new digest correctly incorporates these updates. The DDH-based construction supports a similar style of incremental updates. A frequently changing database, however, requires a client to obtain a fresh and correct digest before making each PIR query. One possible solution to this is to use a public log and a timestamping service [STV+16, TD17].

3.6 Implementation

We implemented all of our authenticated-PIR schemes in roughly 4k lines of Go and 45 lines of C. Our function-secret-sharing implementations are based on the Function Secret Sharing (FSS) Library [Wan17]. Our Merkle-tree implementation is based on the `go-merkletree` library [Tec19]. We implemented group operations in our single-server scheme from the DDH assumption with the CIRCL library [FHK19]. The single-server scheme built on the LWE assumption uses a plaintext modulus of 2^{128} and relies on the `uint128` library [Cha22].

We also implemented multi-server *unauthenticated*-PIR schemes as baselines for comparison. The multi-server unauthenticated-PIR scheme, also used in the authenticated-PIR scheme for point queries, is over the binary field and uses `fastxor` [Cha18]. We use the original implementation of SimplePIR [HHCG+23] as our single-server PIR baseline.

Our implementation is available under open-source license at https://github.com/dedis/apir-code.

3.6.1 Privacy-preserving key directory

To evaluate the practicality of authenticated PIR, we built Keyd, a PGP public-key directory service that offers (1) classic key look-ups and (2) computation of statistics over keys. A key-directory service maps human-memorable identifiers, such as email addresses, to cryptographic identities (public keys). Examples of such directories are the MIT PGP Public Key Server [mit], along with the public-key directories that secure-messaging solutions, such as Signal, implicitly offer.

We implement Keyd in the two-server model, where the security properties hold as long as at least one server is honest. The Keyd key service provides the following properties:

- **Privacy**: The client reveals no information to the servers about the content of its query.
- **Integrity**: The client is guaranteed to recover the *correct* result for the issued query, i.e., the output of the protocol is consistent with the honest server's view.

Prior key-server designs ensure only one of these two properties. It is possible to add privacy to a key server using conventional PIR and issue private complex queries using Splinter [WYG+17], *or* to add integrity as in CONIKS [MBB+15]. Prior to authenticated PIR, we are unaware of any approach that simultaneously solves both problems in the presence of malicious servers, without resorting to trusted hardware [Mar17a].

Keyd lays out public keys in the database using a hash table that maps public keys into fixed-size buckets. To retrieve a PGP public key, a client hashes the requested email to determine the corresponding bucket number, queries the servers for the contents of the bucket, reconstructs and validates the answers, and finally selects and outputs the key of interest.

To evaluate a predicate query, the client sends the query to the servers, which apply it to the appropriate PGP key metadata. For example, to evaluate a `COUNT` query on the email addresses, the client sends `SELECT COUNT(*) FROM email WHERE email =` p, where p represents the query parameter hidden through secret sharing. The `AVG` query is implemented using a `SUM` and `COUNT` query. We use TLS to protect the communication

between client and servers.

Our Keyd serves a snapshot of SKS PGP key directory [Tub21] from 24 January 2021. We excluded public keys larger than 8 KiB, a limit that we found excluded only keys with large attachments (e.g., JPEG images). We also removed revoked keys, keys in an invalid format, and keys with no email address in their metadata. We kept only the primary key of each public key. If multiple keys were linked to the same email address, we kept only the most recent key. If a key included multiple emails, we indexed this key using the primary email. As a result, our Keyd serves a total of 3,557,164 unique PGP keys (≈3 GiB in total), which is more than half of the keys in the original dump.

3.7 Experimental evaluation

We experimentally evaluate all of our authenticated-PIR schemes and the Keyd public-key directory service.

Parameters. We instantiate our multi-server authenticated-PIR scheme for predicate queries using $\mathbb{F}_p^4$ with $p = 2^{32} - 1$, yielding a security parameter of approximately 124 bits. This approach is faster than using a full 128-bit field element, because of better-optimized libraries and CPU instructions for operating on 32-bit values. The Merkle-based scheme for point queries uses BLAKE3 as the hash function. The DDH-based single-server scheme (Section 3.5.2) uses the P256 elliptic curve as the group. We select the parameters for the LWE-based schemes (Section 3.5.1) to ensure 128-bit of privacy according to current estimate of concrete security against known attacks [APS15]. We present one scheme with integrity error 2^{-128}, and another one that uses integrity amplification (Section 3.3.3 and Construction 5), with integrity error 2^{-64}. The scheme with integrity error 2^{-128} uses modulus $q = 2^{128}$ and lattice dimension $n = 4800$; the scheme with integrity error 2^{-64} works with $q = 2^{32}$ and $n = 1100$. For both implementations, the error distribution is the discrete Gaussian distribution with standard deviation $\sigma = 6.4$. As the base authenticated PIR scheme for integrity amplification (denoted PIR_0 in Construction 5) we use the LWE-based scheme (Construction 3) with modulus $q = 2^{32}$ and lattice dimension $n = 1100$, which has integrity error $\epsilon = (2B - 1)/(q - 4B + 1)$ (Theorem 73). The correctness of Construction 3 (Theorem 71) states that $B \leq \sqrt{\lambda N} s$. By Theorem 63 we know that if we use a simple repetition code (which corrects up to t errors by expanding each database bit into $2t+1$ codeword bits) and PIR_0 has integrity error ϵ, then Construction 5 has integrity error ϵ^{t+1}. Table 3.2 shows the choice of t to achieve integrity error ϵ_Π in Construction 5 for different database sizes N, where N indicates the number of single bit records in the database.

Experimental methodology. We perform all the experiments on machines equipped with two Intel Xeon E5-2680 v3 (Haswell) CPUs, each with 12 cores, 24 threads, and

Database size N [bits]:	2^{13}	2^{23}	2^{33}
Integrity error $\epsilon_\Pi = 2^{-64}$	3	4	7
Integrity error $\epsilon_\Pi = 2^{-128}$	6	9	15

Table 3.2: Selection of the error correcting code parameter t for different database sizes and integrity errors.

operating at 2.5 GHz. Each machine has 256 GB of RAM, and runs Ubuntu 20.04 and Go 1.17.5. Machines are connected with 10 Gigabit Ethernet. In the experiments for the multi-server schemes and Keyd (Sections 3.7.1, 3.7.2 and 3.7.4), the client and the servers run on separate machines. For single-server schemes we use a single machine that runs both client and server, as the single-server schemes are inherently sequential. We always report the time elapsed from query computation to record reconstruction as user time and the cumulative bandwidth from and to the server(s) as bandwidth. We execute all experiments 30 times and report the median result across executions. We run all the experiments using a single core for each physical machine. For consistency across experiments, we always download the same public-key when evaluating Keyd. We have published our experimental code and results in our source-code repository (see Section 3.6).

3.7.1 Multi-server point queries

Figure 3.2 presents user time and bandwidth overhead for our authenticated-PIR scheme for point queries, compared to classic unauthenticated PIR. Both user time and the bandwidth overheads increase with database size, as each database record includes and additional Merkle proof of size $O(\lambda \log N)$, which the client must fetch and verify. We measure a maximum overhead of 3× for user time and of 1.6× for bandwidth.

Figure 3.3 illustrates how the number of servers affects user time and bandwidth. As all servers answer in parallel, the user time increase is almost negligible. For authenticated PIR, the increase stems from the verification of Merkle proofs. Bandwidth increases linearly for both schemes, since each server receives a query and sends an answer. The absolute bandwidth reported in Figure 3.3 is significantly higher than that in Figure 3.2, as the latter uses a state-of-the-art PIR scheme based on distributed point functions [GI14, BGI15, BGI16] as both unauthenticated scheme and the underlying PIR scheme for the authenticated version. Additionally, the servers represent the database differently: Figure 3.2 uses a vector representation (one data block per database row), while Figure 3.3 uses a matrix representation (multiple blocks per row), balancing download complexity with query complexity.

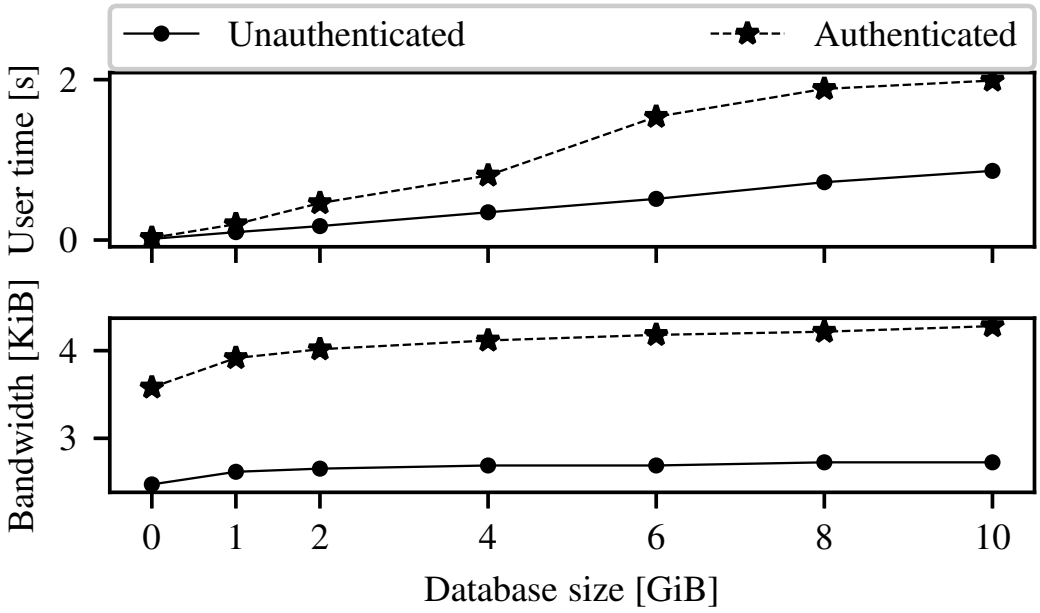

Figure 3.2: The cost of retrieving a 1 KiB record using classic ("Unauthenticated") and authenticated PIR for point queries (§3.4.1) from two servers. For this experiment we use a classic PIR scheme based on distributed point functions [GI14, BGI15, BGI16]. The Merkle proof attached to each record imposes the bandwidth and user time overheads.

In our scheme for point queries, the servers must compute a Merkle tree over the N database entries along with their indexes. The computational complexity of the preprocessing phase is dominated by the number N of database records. Figure 3.4 shows the CPU time that a single server takes to compute a Merkle tree for different database sizes. The current implementation is not parallelized, but in practice, the Merkle-tree computation can be efficiently divided into multiple cores.

3.7.2 Multi-server complex queries

When comparing our multi-server authenticated-PIR scheme for complex queries with classic PIR (Figure 3.5), we find that both the user time and bandwidth overheads of the authenticated scheme are less than 1.1×. The former comes from the longer output of the function-secret-sharing evaluation function—one $\mathbb{F}_{2^{31}-1}$ element versus five elements—and from the verification of the servers' answers, absent in the unauthenticated scheme. For bandwidth, the only difference is the so-called correction word in the function-secret-sharing key [BGI15, BGI16], which is composed of a single field element in classic PIR and of five elements in authenticated PIR: one for the predicate evaluation's result and four for authentication. The servers' answers have the same ratio: a single field element in the

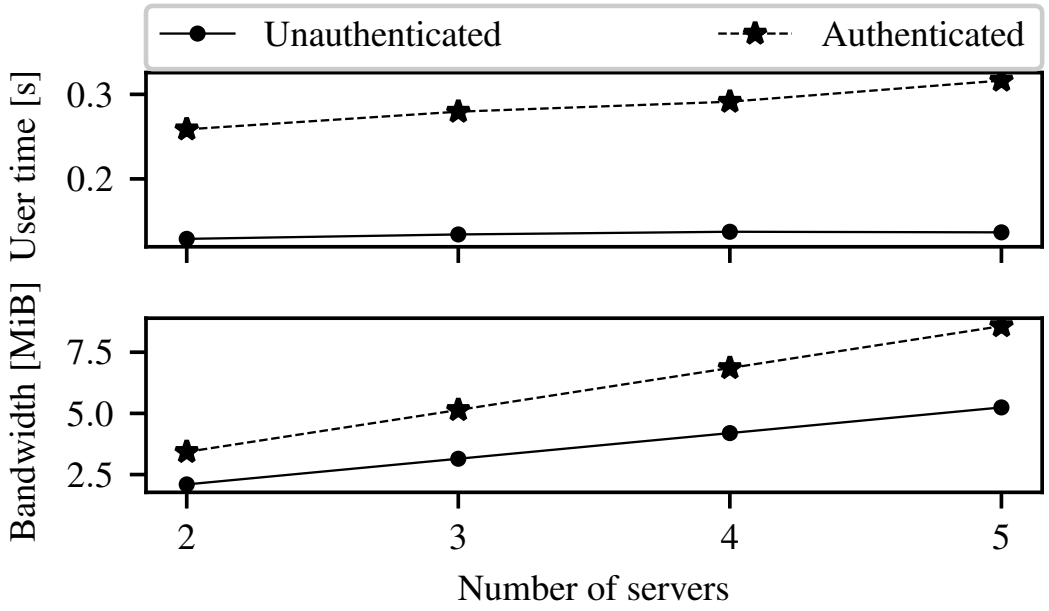

Figure 3.3: The cost of retrieving a 1 KiB record using unauthenticated and authenticated PIR for point queries (§3.4.1) from a variable number of servers holding a database of 1 GiB. For this experiment we use a classic PIR scheme based on classic secret-sharing over the binary field, as schemes based on distributed point functions impose a query bandwidth exponential in the number of servers and database length [BGI15].

unauthenticated scheme and five elements in the authenticated scheme. The bandwidth overhead is thus of a constant factor. Evaluation with $k \geq 3$ servers is infeasible as the length of the keys is $\mathcal{O}\left(\lambda 2^{k/2} 2^{\ell/2}\right)$, where ℓ is the input size in bits [BGI15].

3.7.3 Single-server point queries

To evaluate our single-server authenticated-PIR schemes, we compare their performance against SimplePIR [HHCG$^+$23], the fastest classic single-server PIR scheme for small records to-date. We measure the costs of retrieving one data bit from the database.[2] We evaluate SimplePIR with its default configuration of 2048-bit database records. The client downloads a corresponding record and selects a desired bit from it. The offline bandwidth indicates the digest for authenticated schemes, and the hint for SimplePIR, as this scheme is a PIR-with-preprocessing scheme [BIM04]. We show the results in Figure 3.6.

[2]Other recent PIR schemes (e.g., [MCR21, MW22]) are competitive only in the large-record setting (where records are tens of kilobytes long).

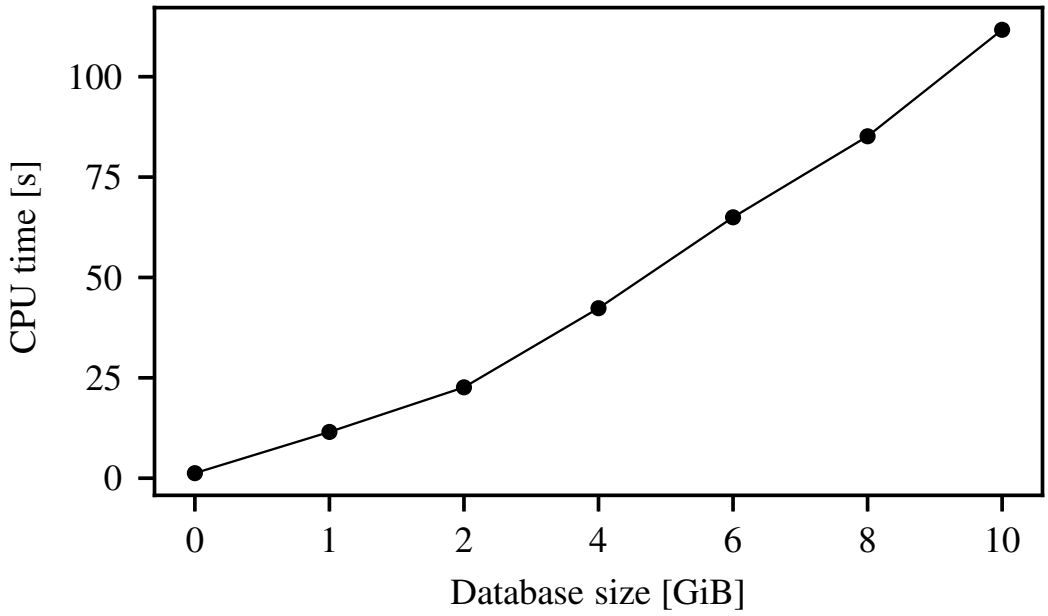

Figure 3.4: The CPU time that a single server takes to process the database for the authenticated PIR scheme for point queries (Section 3.4.1). The Merkle-tree computation is not parallelized.

The authenticated-PIR schemes from the decisional Diffie-Hellman assumption (DDH) and from the learning-with-errors assumption (LWE) have integrity error 2^{-128}. The DDH construction has a smaller digest, hence lower offline bandwidth, but has twice the online bandwidth of the LWE construction: both have the same asymptotic complexity, but LWE uses elements from $\mathbb{Z}_{2^{128}}$ and DDH from the elliptic curve P256, which encodes elements in 256 bits. The LWE construction is also faster (3-79×): arithmetic computations in $\mathbb{Z}_{2^{128}}$ are faster than elliptic-curve operations in P256.

The scheme with integrity amplification (LWE$^+$) has integrity error 2^{-64} and the same classic-PIR privacy as SimplePIR, except that SimplePIR does not provide privacy under selective-failure attacks. LWE$^+$ is faster than LWE for the 1 KiB and 1 MiB databases, but slower (1.4×) for the 1 GiB database: the repetition code requires repeating the protocol 15 times ($t = 7$). An error correcting code with higher rate, or parallel execution of the repetition code, could improve LWE$^+$. SimplePIR is 30-100× faster than LWE$^+$ due to its preprocessing for reducing online computation and exploiting a faster database representation through packing [HHCG$^+$23]. The asymptotic online and offline bandwidth overhead of SimplePIR and authenticated-PIR schemes from the LWE assumption are the same, but integrity amplification increases online bandwidth by $2t + 1\times$ (Section 3.3.3), whereas the client must download the digest only once. Concrete offline bandwidth is lower in SimplePIR due to database packing.

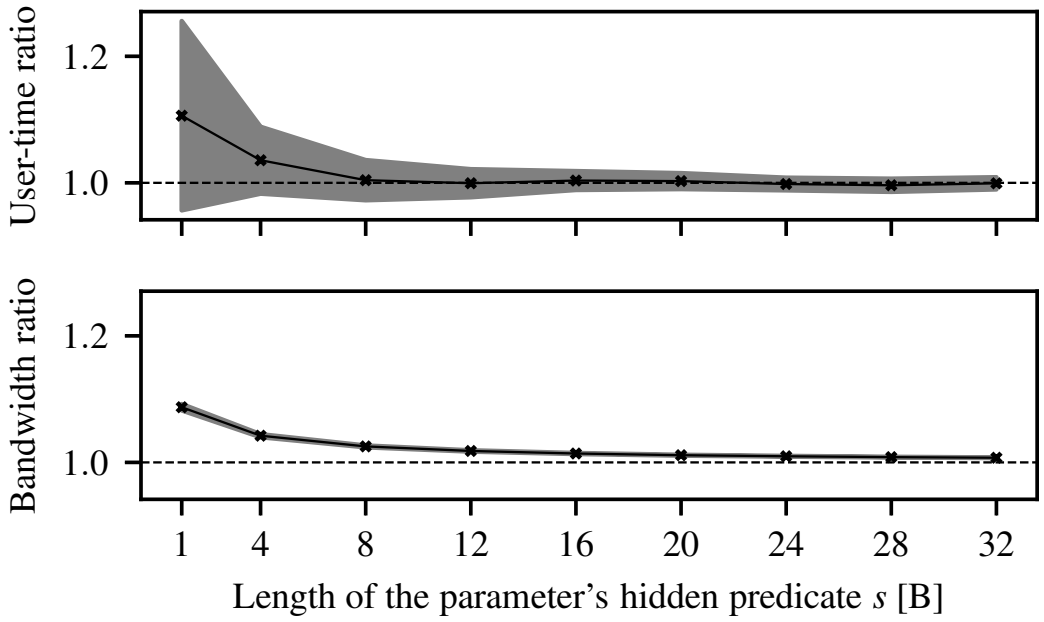

Figure 3.5: The user time and bandwidth ratios between unauthenticated and authenticated PIR (Section 3.4.2) for complex queries when querying two serves for the query `SELECT COUNT(*) FROM keys WHERE email LIKE "%`s`"` from a database composed of 100,000 random records. The median authentication overhead is less than 1.1× for both user time and bandwidth; the grey area shows the variance.

The current schemes are computationally costly, but we expect that future optimizations, such as multi-bit queries, as outlined in Appendix B.4.2, could reduce this cost.

3.7.4 Application: privacy-preserving key server

In this section, we evaluate our multi-server authenticated-PIR schemes in the context of the Keyd public-key server.

For classic key look-ups, which are point queries, we measure the wall-clock time needed to retrieve a PGP public-key with authenticated PIR (Section 3.4.1), classic PIR without authentication, and by direct download without privacy protection. To measure the latency of direct download, we download a PGP public-key from the OpenPGP key server using `wget`. Both PIR measurements include a manually-added RTT of 0.4 ms (the ping time to the nearest PGP key server). We perform all the measurements over the entire processed dataset of PGP keys (see Section 3.6). We measure 1.11 seconds for authenticated PIR, 1.10 seconds for unauthenticated PIR and 0.22 seconds for non-private direct look-up.

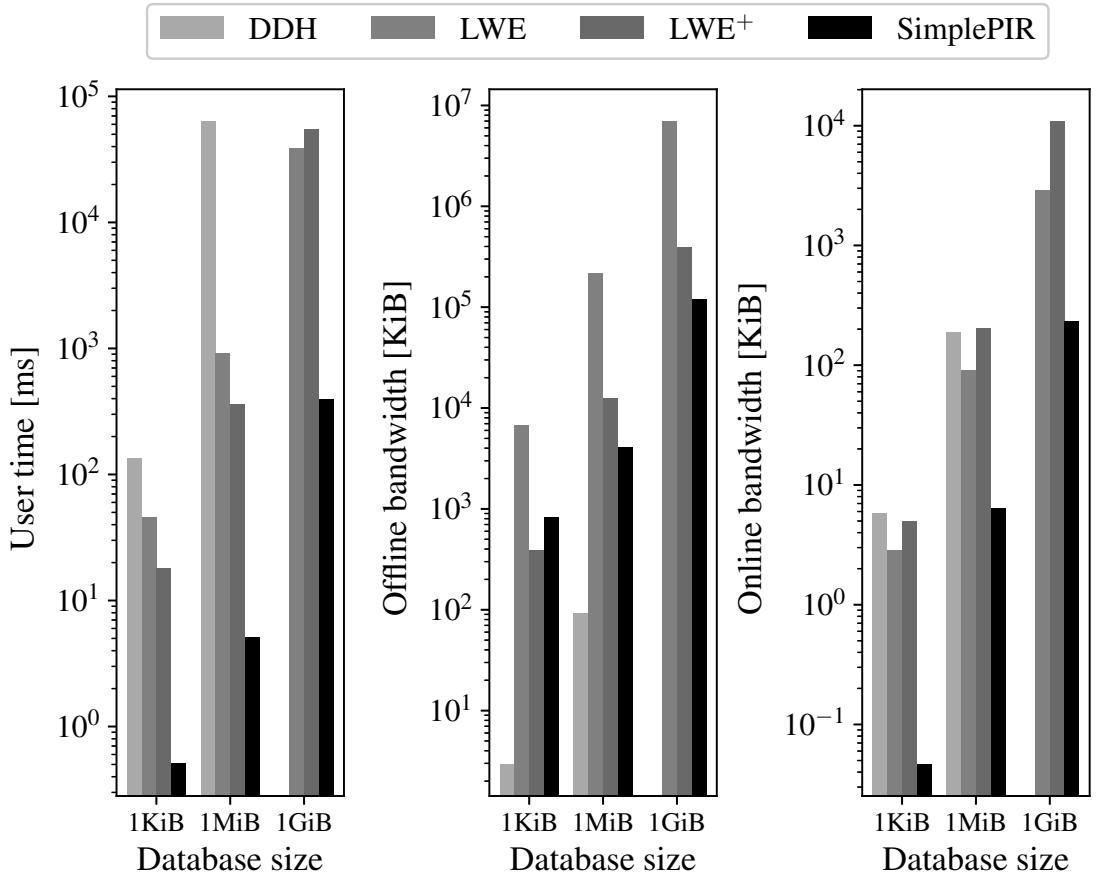

Figure 3.6: The cost of retrieving one data bit using our single-server authenticated PIR schemes and state-of-the-art classic single-server PIR scheme SimplePIR [HHCG+23]. DDH indicates Construction 4 with 2^{-128}-integrity; LWE indicates Construction 3 ($q = 2^{128}$) with 2^{-128}-integrity; LWE$^+$ indicates Construction 5 (the base scheme is Construction 3 with $q = 2^{32}$) with 2^{-64}-integrity (see Section 3.3.3). DDH takes over an hour to retrieve a data bit from a 1 GiB database and we omit it from the figure.

The authenticated scheme for point queries shows performance comparable to classic PIR without authentication. The Merkle-proof overhead in this case is smaller than in Figure 3.2 due to a larger block size and hence less authentication data per data bit in Keyd. The OpenPGP key server maintainers informed us that their service typically handles around 3–10 public-key lookups per second, or less than 1 million requests per day [Bre21]. A careful multithreaded implementation of our multi-server authenticated-PIR schemes for point queries can handle this load with 12 cores, just one more than the number of cores estimated for classic unauthenticated PIR (11 cores).

To analyze Keyd's performance in computing private statistics over keys, we measure user-perceived time and bandwidth of different predicate queries. Table 3.3 shows the results.

Query description	User time [s]			Bandwidth [KiB]		
	Unauth.	Auth.		Unauth.	Auth.	
`COUNT(*) WHERE email LIKE '%.edu'`	25.77	25.97	1.01×	1.8	1.9	1.06×
`type = 'ElGamal'`	7.52	7.66	1.02×	0.9	1.0	1.11×
`YEAR(created) = 2019 AND email LIKE '%.edu'`	48.28	48.32	1.00×	3.0	3.1	1.03×
`AVG(lifetime) WHERE email LIKE '%.edu'`	25.74	26.59	1.03×	1.8	1.9	1.05×

Table 3.3: Performance of different predicate queries on Keyd for unauthenticated and authenticated PIR (the two-server schemes for predicate queries). The median authentication overhead is 1.01× for user time and 1.05× for bandwidth.

For all the predicates, the overhead of authenticated PIR—in both user-perceived time and bandwidth—is upper bounded by a factor of 1.05×. This result matches the benchmark presented in Figure 3.5 and is due to the latency being dominated by the function-secret-sharing evaluation, which is essentially equal for authenticated and unauthenticated PIR. For bandwidth overhead, the same reasoning as in Section 3.7.2 applies.

3.8 Related work

Authenticated PIR builds on diverse work on private information retrieval. Starting with the original proposal [CGKS95], improvements have reduced the communication cost of multi-server PIR with information-theoretic [BI01, BIKR02, WY05, Yek07, DG16] or computational security [CG97, BGI16]. Kushilevitz and Ostrovsky [KO97] presented the first single-server PIR construction, and subsequent work reduced communication costs [CMS99, Lip05, GR05, OS07, DGI+19]. Recent advances introduced PIR for more complex (e.g., SQL-like) queries [OG10, RPG07, WYG+17].

Kushilevitz and Ostrovsky [KO97] first noted that, in the single-server setting, the server could violate a client's privacy by manipulating database records and observing whether the client accepted the response as valid. Such attacks have come to be known as *selective-failure attacks* [KS06, LP11, HKE13]. To our knowledge, we are the first to address selective-failure attacks in the multi-server setting.

In schemes that resist faulty servers (summarized in Table 3.1), a client can either *reconstruct* the correct database entry, or can *detect and abort*, when servers misbehave. Multiparty computation literature refers to the former approach as "full security" and the latter as "security with abort" [GMW87].

Beimel and Stahl [BS02, BS07] first consider malicious or crashing servers in the multi-

server setting. Their approach focuses on ensuring data reconstruction, not detection of server misbehaviour, and it is further developed by concurrent and follow-up work [YXB02, Gol07, DGH12, Kur19, EKN22]. Unlike authenticated PIR, these approaches require an honest majority in the presence of malicious servers, with specific thresholds shown in Table 3.1.

Verifiable PIR in the multi-server setting [ZS14] offers security properties similar to authenticated PIR, but requires expensive public-key cryptography. In the single-server setting [WZ18, ZWH21], verifiable PIR is not resistant to selective-failure attacks and offers a weaker property: it ensures that the server answer a query with respect to *some* database, but not necessarily the one intended. Our approach ensures that queries are answered with respect to a *specific* database, as determined by the honest server in the multi-server setting, or by the database digest in the single-server case. In concurrent work, Ben-David et al. [BDKP22] introduce another notion of verifiable PIR in the single-server setting, whose goal is to verify arbitrary properties on databases, but they do not consider selective-failure attacks.

Our multi-server scheme for point queries (Section 3.4.1) extends a Merkle-tree approach by Kushilevitz and Ostrovsky [KO97]. Our multi-server scheme for predicate queries builds on function secret-sharing [BGI15, BGI16, dCP22, BCG$^+$21], information-theoretic message authentication codes [CDF$^+$08], and malicious-secure multiparty computation protocols [BDOZ11, DPSZ12].

Prior systems address integrity in private information retrieval [DFL$^+$20, MOT$^+$11], but do not protect against selective manipulation in the single-server setting, and require additional assumptions in the multi-server setting.

Prior work has also considered privacy-preserving and integrity-assuring key directories [MBB$^+$15, MPC$^+$18, CDGM19, TFBT21, CDG$^+$22]. In particular, Melara et al.'s CONIKS [MBB$^+$15] and its improved version SEEMless [CDGM19], ensure consistency for the bindings thanks to ideas adapted from transparency log systems [Lau14, Rya14], but do not address privacy of the client's queries.

3.9 Conclusion

Authenticated PIR enhances the strong privacy properties of classic PIR with strong data-authentication guarantees. We have introduced formal definitions both in the dishonest-majority setting—where the security properties hold as long as at least one of the server is honest—and in the single-server setting. We implemented and evaluated all the schemes that we introduced, demonstrating the practicality of multi-server authenticated PIR through Keyd, a privacy-preserving PGP key directory that ensures both the privacy and integrity of retrieved public keys.

4 Real-world deniability in messaging

In this chapter we explore real-world deniability in messaging. We propose a formal model that considers the entire messaging system to analyze deniability in practice. Applying this model to the Signal application and DKIM-protected email, we demonstrate that these systems do not offer practical deniability guarantees. Additionally, we analyze 140 court cases in Switzerland that use conversations on messaging applications as evidence and find that none consider deniability, providing evidence that this property does not have an impact in the legal setting. Based on these technical and legal findings, we assess whether deniability is a desirable property and the challenges and shortcomings of designing a system that is deniable in practice. We posit that systems should either offer real-world deniability or refrain from claiming to achieve it. We discuss how to choose an appropriate threat model for deniability in a given context and how to design communication systems that are deni- able in practice. For Signal, we propose and discuss a simple yet effective solution: the application should enable direct modification of locally stored messages in the user interface.

A preliminary version of this work was presented at RWC 2023. An extended abstract will appear at PETS 2025 [CCHD25], and the full version is available on the Cryptology ePrint Archive [CCHD23]. This chapter's contributions result from a collaboration with Daniel Collins and Loïs Huguenin-Dumittan. The author of this thesis significantly contributed to the design of the model for real-world deniability, the technical analysis of DKIM-protected email and KeyForge [SPG21], the study of legal cases, and the final discussion.

4.1 Introduction

Deniability, according to `www.dictionary.com`, is "the ability to deny something, as knowledge of or connection with an illegal activity". Despite the negative connotation, this definition captures the coarse notion agreed upon by researchers and practitioners: deniability enables a user to *plausibly deny* their involvement in executing some scheme or protocol. The Signal protocol [MP16, PM16], which is the de-facto standard for secure messaging, claims to offer deniability. Moxie Marlinspike, one of the designers of Signal, discusses deniability in the context of Off-the-Record (OTR) [BGB04] as follows [Mar13]:

> "If someone receives an OTR message from you, they can be absolutely sure you sent it (rather than having been forged by some third party), but can't prove to anyone else that it was a message you wrote."

In the cryptographic literature, deniability is typically formalised as a game played between abstract entities. A protocol is deniable if a *judge* cannot differentiate between a real execution of the protocol and a simulated one. This means that any genuine execution of the protocol could have been faked, thereby not implicating a given party who is possibly being framed. For secure messaging, a judge might need to distinguish between actual cryptographic conversations (a sequence of ciphertexts) and simulated transcripts [RGK06, BFG$^+$22b, RG05]. The judge might actively collude with one or more participants before the entire real transcript has been generated [DKSW09, UG18] and/or have access to the cryptographic secrets of one or more parties [BFG$^+$22b, CHMR23].

However, it is crucial to question whether these models (1) are accurate, and (2) are applicable in the real world, such as in a court of law [Gre14a]. In Section 4.2, we argue that neither of these conditions hold true in the context of secure messaging. For (1), existing models fail to consider the higher-level context in which parties execute the cryptographic protocol (e.g., a client that keeps messages in the device's memory and authenticates to a server), thereby rendering their deniability claims vacuous. For (2), the notions of cryptographic deniability do not align with how a human judge would interpret evidence in practice. While we do not question the *technical* value of prior work on the *cryptographic* nature of deniability, we highlight their limitations when considering practical deniability *in real-world scenarios*. In essence, cryptographic deniability is theoretically sound, but its practical implications are weakened by real-world contexts and human factors that these models do not address.

Motivated by these shortcomings, we propose in Section 4.3 a new model for deniability in secure messaging. Our model captures the fact that, in practice, messages are routed between users via a server that usually *authenticates* users. Consider two parties, Alice and Bob, where Bob incriminates Alice. In our model, the judge receives Bob's state (e.g., their entire phone or screenshots of the conversation) after allegedly communicating with Alice. The judge also has data from the server and any other relevant information available.

The system is deniable if there exists a *practical* simulator who, under *application-specific constraints*, can interact with the server and produce a state that is indistinguishable from Bob's. This approach extends the classical notions, which only consider the cryptographic transcript, by providing the judge with Bob's state, which can include his entire phone, a portion of the server's state and arbitrary auxiliary data [RGK06]. Our model broadens the purely cryptographic approach incorporating real-world evidence and higher-level components that can undermine deniability. By doing so, we aim to provide a more comprehensive model that better captures practical aspects of deniability in messaging systems. We note that, depending on what information the judge is afforded, conflicting conclusions can be drawn, and therefore care must be taken when modelling and assessing deniability.

In Section 4.4 we discuss how Signal is *not* deniable in practice by distinguishing two cases: normal authentication and authentication with sealed sender—a feature that hides a message's sender from the relaying server [Lun18]. The first approach is not deniable as the Signal server authenticates the sender, whereas the sealed-sender feature fails to provide deniability because every message includes a sender-specific certificate. We also examine related work proposing solutions for the deniability of email with DKIM protection [CHK11, SPG21]. In this context, the role of email domain servers—another form of relying server—also poses challenges to practical deniability. For both Signal and email with DKIM protection, we show how they can be captured in our model and how it identifies the limitations of deniability's applicability.

In Section 4.5 we analyze 140 legal cases in Switzerland that uses WhatsApp conversations as evidence. Since WhatsApp uses the same core protocol as Signal for two-party messaging, these conversations are cryptographically deniable. We find that (1) in only two cases the legitimacy of such evidence is questioned, (2) judges always accept this evidence, even if disputed, and (3) no case mentions or considers deniability. Although our findings cannot be generalized to other countries, our analysis, along with a similar one in the United States [YGS23] shows that cryptographic deniability does not hold up in a legal setting. We also highlight additional results that show how the very idea of deniability is largely unknown in the legal world.

Both technical and legal analysis show that *cryptographic deniability is ineffective in the real world*. In Section 4.6 we discuss whether deniability should be a goal of messaging solutions by analyzing the issues and shortcomings that practical deniability brings. Given our model to analyze deniability and the analysis of technical and legal limitations, we claim that deniability should either not be a goal of messaging solutions or these must aim for *practical real-world deniability*. We argue that for deniability to be practical, it must be *easily accessible to all users*. Under our notion, Signal would achieve deniability if the application allowed users to *modify, insert or delete messages stored on their devices*, enabling *all* users to simulate conversations *in practice*. This approach provides concrete guarantees since, as our legal analysis shows, screenshots or compromised phones are

generally considered authentic [RMA+23, YGS23], and are likely more tangible to a judge than an abstract simulator. If deniability is a goal, we advocate to implement message modification on the local device, in Signal and other secure messaging applications. We also discuss the risks that such editing capabilities entail.

This chapter contains several opinionated claims about deniability. Our goal is to discuss deniability in the real world and to stimulate further discussion on this subject within the privacy-enhancing technologies community and beyond. We acknowledge the efforts made by Signal to provide a highly secure and private messaging solution, for example by not storing any user information metadata [Sig21a, Sig21b], aiding deniability in practice. Our aim is to push these boundaries even further. Similarly, we recognize the work that both researchers and practitioners have done to improve the security of messaging systems and their deniability.

4.1.1 Summary

To summarise, we make the following contributions:

- We propose (Section 4.3) a model to analyze real-world deniability in messaging.
- In Section 4.4.1 we analyze the Signal application and show that it does not provide adequate deniability in practice.
- In Section 4.4.2 we examine solutions [SPG21] for deniability of email with DKIM protection and show that they fail to provide deniability in some scenarios.
- We analyze in Section 4.5 140 Swiss court cases that use WhatsApp chats as evidence, finding no case that considers deniability.
- We argue that either deniability should be set aside as a security property or made practical. For Signal we propose a drastic solution: users should be able to modify all messages stored on their device (Section 4.6).

4.2 Background

This section introduces deniability in messaging and discusses previous research on what we call *meta-deniability*, which explores to what extent and how deniability is a desirable property.

4.2.1 Deniability in messaging

We assume two parties that use a secure messaging protocol supported by a public key infrastructure and a logical server that routes messages. Messaging solutions typically

consist of two main components: an initial key exchange and the actual messaging protocol. Signal implements this initial key exchange using X3DH [MP16] or its post-quantum variant PQXDH [KS23][1], and messaging with the Double Ratchet algorithm [PM16]. The latter regularly updates keys to protect communication in the case of state exposure.

In messaging, deniability usually refers to the ability of a party to deny interaction with another party. The conceptual simplicity of deniability conceals multiple nuances, leading to a plethora of definitions (cf. [DNS98, RGK06, BFG+22b, CF11, DFG+13, HW21, HLLC11, VGIK20, RG05, FJ24] for a non-exhaustive list and the work of Brendel et al. for a survey [BFG+22b, Appendix A]).

The usefulness of deniability has been debated before, for example in the context of OTR in 2014 [Hea14]. We discuss this matter in Section 4.6. While some issues and arguments that we mention have been raised before, we aim to provide a more structured, up-to-date and thorough perspective. Below we broadly survey the literature on secure messaging and argue that existing approaches to achieve deniability are not practical.

Two main flavours of deniability appear in the literature: *online* deniability [DKSW09, UG15, UG18] and *offline* deniability. Online deniability refers to settings where the judge can interact with parties *during* the execution of the protocol. In contrast, offline deniability applies when the judge receives information only after all relevant communication has ceased. Unger and Goldberg conjecture the incompatibility of online deniability with asynchronous key exchange protocols like X3DH [UG15]. Moreover, it is questionably applicable to many practical scenarios. For instance, in court, evidence pertains to *past* events. The scenario where a judge actively colludes with a party to frame another seems unreasonable if the judge is honest and futile otherwise, as the judge can anyway rule against the victim. Additionally, if the judge considers current events and can mount a (remote) shoulder surfing attack, human factors invalidate online deniability. The judge can instruct the incriminating party to frame the victim, e.g., by asking a question that only the victim can answer. This suggests an analogy between online deniability and the Turing test [Tur50], but we leave this parallelism for future exploration. We therefore consider offline deniability in our model: the data pertaining to an interaction between parties is presented to the judge *after* the interaction has concluded.

Fischlin [FM15] considers and formalizes relaxed cryptographic notions of deniability such as content deniability (denying that a given message was sent, but not necessarily that an interaction took place) and time deniability, and is thus able to reason about the deniability of messaging systems like OTR [BGB04].

Vatandas et al. [VGIK20] analyze the deniability of the X3DH protocol. They prove that the protocol is offline deniable using knowledge of exponent-style assumptions [BP04]

[1]The latest versions of Signal use PQXDH for new chats initiated after both clients use an appropriate version [Kre23]. Since chats may last for years, we take into account both algorithms.

which seems inherent in their model. Fiedler and Janson [FJ24] show that PQXDH's deniability can be proven secure under the same kinds of assumptions. Both simulators are "non-constructive", i.e., the simulator exists but it is impossible to show how it works, making their practicality unclear.

Despite these impractical assumptions, the works of Vatandas et al. and Fiedler and Janson are the only ones we are aware of, in the context of X3DH and PQXDH, to consider all cryptographic information produced by the protocol that can frame a party to the execution of the protocol itself. Their definitions of deniability consider signed key bundles as auxiliary input given to the adversary in both the real and ideal cases. These signatures seem to have been abstracted away in previous work on deniability [HKKP22, BFG+22b], limiting the guarantees of these deniability notions, as signatures frame parties to protocol executions (as discussed in Section 4.4.2 for DKIM-protected email) and preclude strong forms of deniability [HKKP22, BFG+22b].

Although interesting from a cryptographic perspective, these approaches effectively restrict the transcript to the *cryptographic* material that parties exchange *during the protocol execution*. This limitation affects the *practicality* of deniability, since it ignores a plethora of additional cryptographic and non-cryptographic information that can frame a party to a protocol execution. Even in definitions where auxiliary input is given to the judge, this data is either ignored in the analysis or only instantiated with protocol-level cryptographic material (as for X3DH/PQXDH above), and no framework is provided to reason about the system as a whole. Modern messaging solutions such as Signal rely on a complex ecosystem of communication protocols, cryptographic solutions, and asynchronous multi-device management. This stack provides a modern and user-friendly messaging experience, but comes at the cost of deniability.

4.2.2 Meta-deniability for messaging

This section discusses related work on deniability in messaging from a *user perspective*. Assuming that most current messaging solutions achieve some sort of cryptographic deniability, these works focus on how deniability is perceived in the real world and the missing steps to achieve practical deniability, or *human*, deniability.

Reitinger et al.'s survey study [RMA+23]. In a recent work Reitinger et al. conducted a user study involving US-based participants to investigate the requirements for achieving practical deniability given a messaging protocol featuring cryptographic deniability. Participants acted as a jury in a (role-played) courtroom and were shown a screenshot of a Signal conversation suggesting a politician took a bribe. They had to decide whether the defendant was guilty or not in different scenarios:

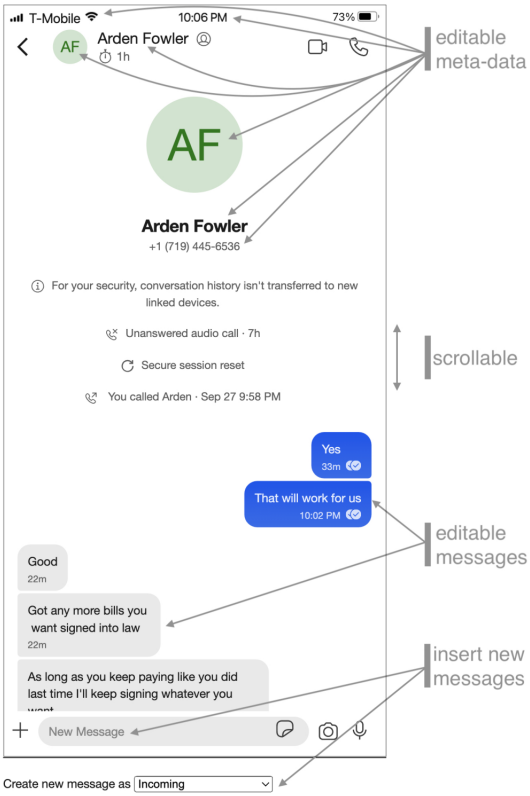

Figure 4.1: Interactive forging tool that Reitinger et al. use for their study of deniability in secure messaging [RMA+23].

1. *No defense*: The defense claims that the screenshot is fake.
2. *Expert*: Renowned cryptographers explain to the jury that the protocol is deniable and therefore there is no proof that the message, and thus the screenshot, is legitimate. The study's authors apply this scenario using two approaches, namely, with jargon or friendly parlance. In the former, the experts argue using technical terms, while in the latter they use less technical and more understandable language for non-experts.
3. *Tool*: Any user of the messaging application can edit anything in the chat, e.g., insert or modify messages or modify delivery times. The authors define three sub-scenarios: (3.1) the tool exists and the participant is given a fake screenshot; (3.2) anyone can use the tool and the participant gets the same fake screenshot; (3.3) the tool is available to the participant, who can actively modify the screenshot (Figure 4.1 shows the tool that the participant can use).

The study shows that while 71% of the participants would decide *guilty* in the no defense scenario (1), this number drops to at most 26% in the expert (2) or tool (3) cases. Additionally, the study reveals that the active forgery tool is the most convincing

scenario, i.e., participants who can use the tool are significantly more likely to believe that screenshots can be forged. These results show that cryptographic deniability is effectively not accepted or understood. While cryptographers might assume that without cryptographic evidence all accusations must be deniable, this study suggests otherwise: denying authorship of a screenshot does not convince people that it is fake, even in the absence of cryptographic or other evidence. People's perception of deniability does not necessarily align with the formal cryptographic perspective. On the other hand, the study shows that the presence of a practical forging tool increases the likelihood of achieving plausible deniability.

Yadav et al.'s survey study [YGS23]. Yadav et al. conducted a study on deniability by analyzing three components: support of deniability in the messaging application, social acceptance of deniability and legal recognition of deniability. Like the study of Reitinger et al. all the participants of this study were based in the United States.

Their analysis of social acceptance of deniability reveals that participants tended to trust conversations from a messaging application more than oral claims made by a participant in a conversation. This implies that deniability is not socially accepted: if it were, participants would place the same amount of trust in both oral and digital claims. The lack of social acceptance is likely influenced by lack of knowledge about this property. The study shows that only 0.6% of participants accurately interpreted OTR's deniability definition, while 64.8% believed they understood it actually did not. Additionally, 32% of participants found the deniability definition self-contradictory, highlighting confusion and misinterpretation.

The survey indicates that most users prefer non-repudiation, i.e., non-deniable digital communications. Approximately 60% of users desire only non-repudiation, whereas 12.7% and 4.5% prefer deniability or anonymity, respectively. The remaining 22.6% seek some combination of these properties. Participants cited instances where non-repudiation was a requirement (98%) and emphasized its importance (82%), while deniability was less frequently needed (60.94%) or deemed highly important (23.18%).

This study provides evidence that cryptographic deniability has not been considered in US court cases involving WhatsApp chats as evidence. Out of 228 cases analyzed, none presented an argument for cryptographic deniability. Instead, judges demanded concrete evidence rather than accepting claims of message forgery. There is a need for court cases that present valid technical arguments for deniability in real-world scenarios to determine whether deniability will gain legal acceptance.

4.2.3 Additional related work

Celi and Symeonidis [CS20] presented a talk on deniability at HotPETs 2020, with a particular focus on messaging. The authors discussed a number of open questions regarding deniability in messaging, most of which remain unanswered today. To the best of our knowledge, no extended version of this work has been published.

Different works propose methods to balance non-repudiation and deniability in various settings, such as message franking [GLR17, TGL+19, IAV22, BE24] and the recently introduced concept of retroactive avowal [WCWB24].

This work draws inspiration from other research that model legal and real-world scenarios using cryptography. Cohen and Nissim [CN20] formalize aspects of the European privacy law. Garg et al. [GGV20] formalize the right to be forgotten. Frankle et al. [FPS+18] propose using zero-knowledge proofs to enable public audits of warrants issued confidentially by intelligence courts. Scheffler and Varia [SV21] study the guarantees of cryptographic protocols against compelled disclosure through self-incrimination.

4.3 Our model

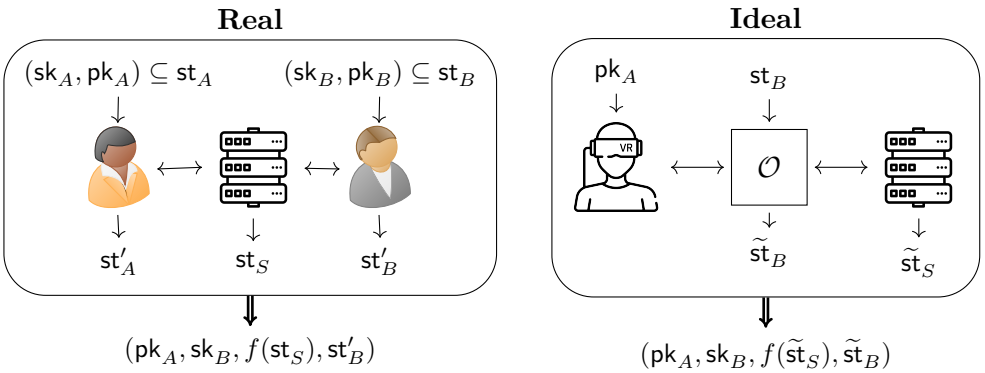

Figure 4.2: Our model for real world deniability in a two-parties protocol. The simulator is depicted by the virtual reality device. The application-specific oracle $\mathcal{O}$ models the ability of the simulator to modify Bob's state. The judge, aided by some auxiliary material aux, must distinguish between the output of the real and the ideal worlds (see Definition 57).

This section introduces our deniability model, which captures deniability *in practice*. We depict our model in Figure 4.2, and adopt the real/ideal paradigm: the judge must differentiate between the ideal and real worlds. Our model captures the interaction between two parties, Alice and Bob, and aims for offline deniability. In this scenario, Bob, the incriminating party, hands over his state to the judge *after* the protocol execution to frame the victim, Alice. We assume that both parties execute the protocol *honestly*,

meaning that Bob does not deviate from the protocol to incriminate Alice. This does not mean that Alice and Bob have to send everything through their own actions: e.g., Bob can induce Alice to send a ciphertext over the wire if the protocol implements read receipts [Wha24a, MKA+21].

In the real world (Figure 4.2, left), Alice and Bob execute the protocol using an external server. The server is a logical entity that can represent several physical machines in a centralized or distributed setting. Both parties input an initial state (st_A and st_B) that contains, for example, their long-term private key and the counterpart's public key. We do not force a type for a party's state: it can be anything, e.g., the entire phone. The real world returns Alice's public key pk_A, Bob's secret key sk_B (which is possibly blank, i.e., $\perp$), a function of the server's internal state $f(\mathsf{st}_S)$, and Bob's final state st'_B. The function f models different types of leakage from the server. For example, a subpoena might force the maintainers to reveal some of the server's data, in which case f outputs only a partial view of the server's data. Alternatively, if the server is completely breached, f is the identity function.

The ideal world (Figure 4.2, right) encompasses three actors: an oracle, a simulator (represented by the person with the virtual reality headset in Figure 4.2) and a server. The server is an exact replica of the one in the real world. The simulator inputs the victim's public key, i.e., Alice's pk_A, and interacts with the oracle to produce a simulation of Bob's final state. The goal of the simulator, in collaboration with the oracle, is to demonstrate that Bob can reach a simulated final state $\widetilde{\mathsf{st}}_B$ starting from an initial state st_B *without interacting with Alice* (i.e., without knowing st_A). The simulated state $\widetilde{\mathsf{st}}_B$ must be indistinguishable (Definition 57) from the real final state st'_B, which Bob returns in the real world. To this end, the oracle inputs Bob's initial state st_B, interacts with the simulator and the server and, outputs $\widetilde{\mathsf{st}}_B$ at the end of the experiment. The simulator does not directly output the simulated state: state evolution is mediated through the oracle, which complies with the simulator's instructions while considering practical constraints.

The ideal world outputs Alice's public key pk_A, Bob's secret key sk_B, some leakage of the server $f(\widetilde{\mathsf{st}}_S)$ for some function f, and Bob's simulated state $\widetilde{\mathsf{st}}_B$. As Definition 57 states, the judge J, with the help of some auxiliary information—that is, any prior or contextual information the judge might have—must distinguish between the outputs of the two worlds.

Definition 57. We say that a system using a server S, coerced to reveal information that the function f represents, is *deniable* if, for any victim A and any incriminating party B, there exists a simulator with access to an oracle $\mathcal{O}$, such that for any judge J with auxiliary information aux, the following holds:

$$\left|\Pr\left[J(\mathsf{aux}, \mathsf{pk}_A, \mathsf{sk}_B, f(\mathsf{st}_S), \mathsf{st}'_B) \Rightarrow 1\right] - \Pr\left[J(\mathsf{aux}, \mathsf{pk}_A, \mathsf{sk}_B, f(\widetilde{\mathsf{st}}_S), \widetilde{\mathsf{st}}_B) \Rightarrow 1\right]\right| \leq \nu,$$

Oracle NO-AUTH(op, payload)

```
1 : if op = get then
2 :   send st̃_B to simulator
3 : elseif op = set then
4 :   st̃_B ← payload
5 : elseif op = forward then
6 :   send payload to server
7 :   send server's answer to simulator
8 : elseif op = finish then return st̃_B
9 : else return ⊥
```

Oracle AUTH(op, payload)

```
1 : if op = get then return ⊥
2 : elseif op = set ∧ Vf(st̃_B, payload) then
3 :   st̃_B ← update st̃_B according to payload
4 : elseif op = forward then
5 :   authenticate to server using st̃_B
6 :   send payload to server
7 :   send server's answer to simulator
8 : elseif op = finish then return st̃_B
9 : else return ⊥
```

Figure 4.3: Two oracles for our deniability model. The NO-AUTH oracle models a system *without* authentication, AUTH models *signature*-based authentication.

where ν is a value that models the *in dubio pro reo* principle: if the judge is *unsure*, to some degree of uncertainty, then they rule in favour of the victim A. The exact value of ν depends on the context in which our definition is used.

Remark 58. Our definition captures honest protocol executions between Alice and Bob. Achieving deniability in the malicious setting is very difficult if not impossible: Bob can circumvent deniability such as with remote algorithm substitution attacks [AQ22] or remote attestation [GPA19]. Remote attestation leverages hardware-based trusted execution environments, which are widely available in today's mobile phones, to produce a publicly verifiable and non-deniable transcript from the execution of a deniable protocol; the victim does not even detect the loss of deniability, as she runs the original deniable protocol.

Remark 59. Unlike many cryptographic models that formally define variable types, our model requires these variables to be defined before analyzing practical deniability. In many cases, the accuser (Bob) can determine his state, e.g., by surrendering his entire phone to the judge. Similarly, Bob and/or the judge often control the function f that

models leakage from the server, e.g., the judge can subpoena the server or the service can disclose useful metadata to the judge. The variables of the model must therefore be defined *before* applying it to a specific system, taking into account the *worst case* that the threat model encompasses.

4.3.1 Model parameters

Our model provides a large degree of flexibility and can capture many different scenarios. To this end, we discuss below how different parameters of Definition 57 may be chosen before describing and assigning variables in two example scenarios. In some cases, there may be some overlap between variables, e.g. Bob's state may contain his secret key.

Alice's public key (pk_A). This should typically correspond to Alice's long-term public key that she registers with some public key infrastructure, for example the Signal server.

Bob's secret key (sk_B). This typically corresponds to Bob's long-term secret key, but if Bob chooses to not disclose this to the judge, it may be empty ($\perp$).

Server state leakage ($f(\mathsf{st}_S)$). As we consider offline deniability, st_S corresponds to the state of the server after the relevant interactions between Alice and Bob have taken place. The leakage function f depends on the what the server is willing or compelled to disclose and what the judge has access to. In a legal setting, for example, WhatsApp discloses significantly more information than Signal [EGS21, Wha24b]. In some settings, this could capture information that has been leaked from the server to, e.g., the general public through whistleblowing or compromise. In some cases, the judge may not have access to any pertinent information from the server. Although our definition is tailored for the case where Alice and Bob are communicating via a server, direct communication could be captured by considering the 'trivial' server that leaks nothing.

Bob's final state (st'_B). As with Bob's secret key, this can depend on Bob's level of cooperation with the judge (or indeed what they are compelled to disclose), and depends on context. For example, this could comprise of Bob's message database, their cryptographic state, the entire contents of their phone or something in between.

Auxiliary data (aux). This corresponds to any additional information that the judge may have access to, e.g., testimonies, or more generally information about Alice (e.g. their public profile), Bob and their conversation.

The judge (J). The judge corresponds to any entity that wishes to evaluate the deniability of a given execution. In a courtroom, the most classic setting, More broadly, this can capture entities like the general public (or particular individuals) who are

determining whether some interaction was real or not (cf. Example 61). In principle, the judge could be anyone.

The oracle ($\mathcal{O}$). The oracle models the simulator's practical effectiveness in simulating Bob's state and communicating with the server: the initial state might be encrypted with some unknown secret key and the server might require the client to authenticate with it. The simulator instructs the oracle to communicate with the server and to modify the initial state; the oracles comply with the instructions by taking into account the practical constraints. In other words, the simulator captures the level of control Bob has over his client. For example, at one extreme, if Bob has rooted his phone, he can extract all secrets and messages contained within, making the oracle very permissive. At the other extreme, if Bob knows nothing about technology and is limited to using the client to try to forge messages, the oracle is much more restrictive.

Figure 4.3 shows two instantiations of the oracle $\mathcal{O}$ which capture many realistic settings: at the top without authentication and at the bottom with authentication. Both oracles input an operation $\mathsf{op} \in \{\mathsf{get}, \mathsf{set}, \mathsf{forward}, \mathsf{finish}\}$ and a corresponding $\mathsf{payload}$ that the simulator wishes to send to the server or to set as Bob's simulated state. The oracle that mimics authentication checks the payload's legitimacy using the verification function Vf before updating the state. The payload, for example, can be a message encrypted with TLS, which is decrypted and stored in the state only if it comes from the correct server. The simulator interacts with the oracle to simulate Bob's state and communicates with the server while enforcing the authentication mechanism.

The AUTH oracle captures the interaction of the vast majority of users who use a given messaging application as intended and execute the protocol honestly. The $\mathsf{NO\text{-}AUTH}$ oracle, on the other hand, captures a technically-savvy or resourceful user who is able to modify their state directly. The behaviour of the oracle also may vary over time, e.g. in the case of KeyForge for DKIM (cf. Section 4.4.2).

Real-world examples

We provide two examples to highlight how the modelling process can occur. They also highlight how the judge's role can change depending on where and against whom deniability must hold; we expand further on this aspect in Section 4.6.2.

Example 60. Suppose Bob accuses Alice of sending a particular message in a classic courtroom setting with a single judge. Alice and Bob communicate using Signal and they do not know each other, i.e., they have never had any contact before. In this case the variables in the model are defined as follows:

- pk_A: Alice's public key in Signal.

- sk_B: Bob's secret key in Signal.
- $f(\mathsf{st}_S)$: Alice and Bob's last connection data to Signal and the data of the creation of Alice and Bob's contacts, all in Unix timestamp.
- st'_B: Bob's phone, since he decides to surrender it to the judge when accusing Alice.
- aux: since Alice and Bob do not know each other, we can assume that $\mathsf{aux} \leftarrow \varepsilon$, where ε represents the empty string.

The judge J is the judge of the courtroom in which Bob accuses Alice. The justification behind these choices of variables is mostly self-evident, except for the choice of f, which we chose as these timestamps comprise the only information that Signal claims to and has previously disclosed in subpoena requests.[2]

Example 61. Suppose Bob takes a screenshot of what he claims to be his conversation with Alice and posts the image on a public bulletin board. The conversation is on Signal and contains some insults that Alice uttered towards Bob. Shortly after, Carol posts a comment to the image where she says that she once received the same insults from Alice. In this case the variables in the model can be defined as follows:

- pk_A: Alice's public key in Signal.
- sk_B: $\perp$, since Bob's secret key is most likely not given to the judge (see below).
- $f(\mathsf{st}_S)$: In this case $f(\mathsf{st}_S) \leftarrow \emptyset$ since no one sent a subpoena to Signal.
- st'_B: The screenshot that Bob publishes on the bulletin board.
- aux: The auxiliary information contains at least the comments that Carol posts on the bulletin board after Bob's screenshot. Additional information might exist, depending on the people acting as judges.

The judge J is in this case the group of people that consult the public board, thereby seeing Bob's screenshot and Carol's comment.

4.3.2 Limitations and pitfalls

Although our model is able to capture many scenarios, it nonetheless has drawbacks. Firstly, the model only directly captures two-party communication, so cannot be readily used for group messaging or in multi-party settings. In addition, it only captures offline deniability and honest executions (which we nonetheless argue above is reasonable).

[2] See https://signal.org/bigbrother/.

Our model leaves many parameters and variables undefined. Whilst this allows it to capture a variety of settings, care must be taken when it is used. In particular the model requires domain knowledge and fine-tuning to be used effectively, but we see this as inherent in assessing deniability in the real world. Whilst the model is not fully formal, it nonetheless provides a framework for thinking and reasoning about deniability in the real world.

Moreover, care must be taken when drawing conclusions from the model. Suppose that Alice and Bob are messaging, and Bob is able to modify his received messages without leaving any trace of tampering from the perspective of the judge (e.g., by modifying the set of stored messages on another device and then syncing). Then, Bob has successfully framed Alice, and so the conclusions drawn from the model in the real world would be incorrect (even if they are logically consistent). Thus, when using the model, one must ensure that the variables given to the judge are as numerous as possible. In this example, additional auxiliary data or server leakage could catch Bob's tampering, or alternatively ensuring that Bob's state is comprehensive (as here he excluded critical information). If, due to contextual constraints, that some information cannot be feasibly given to the judge, then it is important to be cautious and determine what kind of additional information *could* exist, and what impact that would have on deniability. For this reason, we advocate for the *in dubio pro reo* principle to be applied when interpreting deniability in our model, as it is usually done in the legal setting.

Finally, our model focuses on the cryptographic transcripts and network traces that the interaction between Alice and Bob generates. The oracles that we introduce in Figure 4.3 model the simulator's practical effectiveness in terms of authentication, but do not consider side channels or residual data that might remain on a device after a conversation has taken place. Such residual traces might include deleted data lingering in system I/O buffer, temporary storage or system logs that the operating system or applications maintain. Addressing this limitation requires defining new oracles that explicitly consider side channels and local device artifacts. Looking forward, it is crucial to account for this limitation when evaluating our proposed solution for achieving real-world deniability—namely, local message database modification (Sections 4.4 and 4.6). We leave as an open problem to define such oracles and determine how to ensure that database modifications do not leave exploitable traces that could enable a judge to frame Alice by distinguishing between the real and ideal worlds. Future research could build on related work that examines side channels in encrypted databases and searchable encryption [CGPR15, GSB$^+$17, GRS17]. We further discuss these directions for future work in Chapter 5.

4.4 Technical case studies

This section analyzes two real-world communication solutions and their deniability properties: the Signal application and KeyForge, the latter being system that achieves a form of deniability for email communication while maintaining the protections that DKIM provides [SPG21]. We describe both systems and we apply our deniability model (Section 4.3) to them. In departing from messaging to discuss another communication medium, we show that practical deniability remains hard to achieve even with practical countermeasures against non-repudiation.

4.4.1 The Signal application

We focus on Signal because it offers the best security and privacy guarantees among deployed secure messaging solutions. Moreover, the cryptographic algorithms underlying Signal are deniable under some notions of cryptographic deniability [VGIK20, MP16, PM16] and much of the code is open source. We distinguish two cases: normal authentication and authentication with the sealed sender feature [Lun18]. To conduct this analysis we analyze the Signal server source code [Sig22].

Normal authentication

To send a message, the sender's client issues a POST request to the server's endpoint `/v1/messages/{receiver}` with a payload that contains the encrypted message and the receiver's identifier. The server authenticates the sender using Dropwizard [Dro24] basic authentication, which authenticates the sender's identity and a password (a secret shared with the Signal server). After authentication, the server performs a few extra checks (e.g., rate limiting) and creates an "envelope" that contains the authenticated sender's identity, the encrypted message and a timestamp. The server forwards this envelope to the receiver or stores it until the receiver is online.

Deniability in practice. The description above implies that a message *delivered* by a legitimate and honest Signal client originates from the claimed sender, if the Signal server operates honestly. Thus if someone inspects the receiver's device, they can be confident that the message indeed comes from the sender. If the server is honest and not compromised, the only deniability argument for the sender is to argue that the receiver tampered with the stored messages, i.e., the receiver edited the received messages in the local database. Depending on the device, modifying the database can be challenging (e.g., it requires basic technical knowledge on the desktop client as the database encryption key is stored in clear[3], while advanced knowledge is required for mobile applications). Alice's

[3]Joshua Lund from Signal comments on this on the SignalCommunity forum: "The database key was never intended to be a secret. At-rest encryption is not something that Signal Desktop is currently

deniability claim depends therefore only on Bob's technical knowledge. Depending on the situation, Bob can deny having the ability to tamper with the local Signal database and, in general, this argument is not equally available to every potential victim.

Model variables. To capture Signal, variables in our model can be set as follows.

- pk_A: Alice's long-term public key in Signal.
- sk_B: Bob's long-term secret key or $\perp$, depending on the power of the judge.
- $f(\mathsf{st}_S)$: $\emptyset$ or two timestamps (depending on whether a subpoena was sent to Signal or not, cf. Example 60). This assumes that the Signal server behaves honestly.
- st'_B, aux and the judge: These are highly context-dependent, depending on how much Bob cooperates, how much information the judge has access to, and who or what exactly the judge is, respectively.
- Oracle $\mathcal{O}$: Depending on Bob's (technical) ability to modify his state, either NO-AUTH (if he can do so) or AUTH (if he cannot). In particular, AUTH captures the fact that Signal uses authentication as described above.

This also applies when considering sealed sender in the text below.

In our model

If there is authentication (AUTH oracle in Figure 2.2), Signal is not deniable as Bob's state (comprising the received messages) is updated only upon receiving a message from the Signal server that certifies Alice sent it. As described above, this occurs only if the server authenticates Alice before she sends a message, which the simulator cannot do as it does not know Alice's credentials. In particular, in the AUTH oracle in Figure 2.2, the verification function on line 2 fails because payload does not contain the necessary credentials, i.e., Alice's, to update Bob's simulated state $\widetilde{\mathsf{st}}_B$. This results in two different states st'_B and $\widetilde{\mathsf{st}}_B$, giving the judge a distinguishing advantage of probability one in Definition 57. Note the judge requires no auxiliary information for this attack to succeed.
We formalize Bob's modification of the local database using the same AUTH oracle. The payload that the simulator provides to the oracle includes the database encryption key and the verification function Vf returns true, enabling the oracle to modify Bob's simulated state, i.e, the messages' database.
If the oracle NO-AUTH is used, the simulator can modify Bob's state to make it

trying to provide or has ever claimed to provide. Full-disk encryption can be enabled at the OS level on most desktop platforms" (https://community.signalusers.org/t/vulnerabilities/4548/7). On Linux, the database encryption key is stored in `~/config/Signal/config.json` and the database in `~/config/Signal/sql/db.sqlite`.

consistent with some messages received from Alice. Hence the deniability claim relies solely on the receiver's ability to tamper with the local database (barring any incriminating evidence otherwise available to the judge).

A malicious client can prove that they indeed received the message from a Signal server by using DECO [ZMM+20], which enables clients to prove that data received via TLS originated from a particular server. For this and other reasons, which we present in Section 4.3, we exclude malicious clients from our analysis and deem as impossible deniability in this setting.

Authentication with sealed sender

Authentication with sealed sender conceals the identity of the sender from everyone except the receiver, *including* the Signal server. This protects the sender's identity at the *application level*, but does not hide other sensitive information, such as IP addresses. By default, this option is only enabled between mutual contacts and works as follows. Periodically, clients retrieve a certificate signed by Signal that contains (1) the client's identity, (2) the client's phone number and (3) an expiration date. In addition, each party derives a *delivery token* from their profile key and registers it with Signal. To send a message, the sender's client encrypts the certificate, the message and the sender's identity. The client forwards the resulting ciphertext, the receiver's identity, and the receiver's delivery token to the server using the same endpoint as in normal authentication. The server checks that the token corresponds to the intended recipient, and forwards the ciphertext when the receiver is online; the server does not authenticate the sender. Finally, the receiver decrypts the ciphertext and verifies that the certificate is valid and corresponds to the claimed sender's identity.

Deniability in practice. This setting offers brighter prospects for deniability than normal authentication. As the server never authenticates the sender, receiving a message from Alice that was correctly forwarded by the server does not mean she actually sent it. However, the fact that the message includes the sender's certificate hinders their capability to deny. Let's say the victim Alice claims that Bob (1) forged a message coming from her, then (2) made the Signal server forward that message to himself and finally (3) his client successfully delivered it. It means that Bob managed to recover one of Alice's certificates that is valid for that specific time frame. In turn, that implies that he (or an accomplice) recently received a message from Alice and extracted the certificate to forge a valid message. Therefore, in practice, one can be very confident that Alice exchanged messages with Bob recently, which weakens plausible deniability. Alice can still deny that she sent a particular message, but the claim relies solely on the fact that Bob had the technical knowledge to extract the certificate and forge a valid message or, as in normal authentication, the capacity of tampering with the messages stored on his client.

In our model

Transposing this argument in our model, we see that Signal with sealed sender is not deniable in the authenticated setting ($\mathcal{O} = \mathsf{AUTH}$) as Bob's state will accept Alice's message only if a legitimate certificate was provided. However, the simulator cannot obtain Alice's certificate from the server as it authenticate as Alice: the verification function Vf returns false and Bob's state is not updated. In the setting where Bob can freely modify the set of received messages ($\mathcal{O} = \mathsf{NO\text{-}AUTH}$), Signal with sealed sender is deniable.

Summary and discussion

The above discussion shows that plausible deniability is practically non-existent in Signal unless the receiver has good technical capabilities. To summarise:

- Using normal authentication, if the client delivers a message from Alice, then it surely came from Alice because the server authenticated with her.
- Using sealed-sender-based authentication, if Bob's client delivers a message from Alice, then with good probability Bob received a message from Alice recently. Bob could forge a message coming from Alice by asking for Alice's certificate from one of Alice's contacts, but we deem this to be relatively impractical.
- In both modes of authentication, deniability is plausible only if the receiver has good technical knowledge to modify the local database of received messages, i.e., to modify the state of the incriminating party ($\widetilde{\mathsf{st}}_B$ in our model). If the incriminating party lacks technical knowledge, the judge will hardly believe the victim's denial of participation.

We conclude that, in most cases, the Signal application does not provide plausible deniability.

It is worth noting that the deniability claim becomes stronger as the system becomes easier to breach, and vice versa. For instance, in a perfectly "secure" system (e.g., impenetrable clients and interactions only through authenticated clients as in our model with $\mathcal{O} = \mathsf{AUTH}$), Signal is not deniable. This creates an undesirable situation: certain aspects of security clash with deniability. We propose that the best way to address this problem is to either give up on deniability or to incorporate "cracking" as a controlled and desirable feature. With cracking we refer to the usually illicit act of circumventing or overcoming security measures, such as accessing the protected Signal's database and modify its content, or compromising Signal's server to send a message under a spoofed identity. For example, Signal could include an edit/add button that allows the user

to modify received messages or add new ones. The user interface could look like the interactive forging tool that Reitinger et al. used for their study (Figure 4.1). While this could lead to other problems as discussed in Section 4.6, we claim that if we want deniability in Signal and other messaging application, then we should aim for human and practical deniability instead of just cryptographic deniability, which, as we show, does not translate into real-world deniability. Overall, this form of deniability can only be guaranteed if a practical, user-friendly simulator exists, which is currently not available. We continue the discussion about how to achieve practical deniability and possible shortcomings in Section 4.6.

4.4.2 DKIM and KeyForge

Another interesting case study for deniability is email-based communication. Here, we assume that emails are not authenticated with PGP or S/MIME: in such cases the email is cryptographically signed by the author and thus the fact that it was sent is publicly verifiable and undeniable.

We focus on email protected with DomainKeys Identified Mail (DKIM) [CHK11], which authenticates the source of an email by verifying the domain name from which it originated. If an email is sent from `first.com` to `second.com`, the server of `first.com` cryptographically signs the email (including message and headers) and adds the signature to the email's headers. When the email reaches `second.com`, the server verifies the signature using `first.com`'s public key and delivers the email only if the verification is successful. If the DKIM signing keys are kept private, the signature proves that the message was not altered during transit and that it was sent by an authorized sender, thereby preventing malicious activities such as spoofing, phishing and spam.

As the leak of Hillary Clinton's emails [Wik16b] shows (see Section 4.6 for further discussion), DKIM protection devastatingly impacts email deniability. The digital signature provides a cryptographic assurance that the sender is who they claim to be, unless email accounts are breached or secret keys are leaked, which we deem unlikely[4]. Using our model (Section 4.3), DKIM-protected email is not deniable since we need an oracle like AUTH to model this setting: the only way to update Bob's state is to have control over the signing keys of all domains between the sender and the recipient, unless another breach occurs.

To remedy this situation, Specter et al. propose KeyForge [SPG21], which enforces that DKIM signing keys are released after a predefined delay, which the authors recommend to set at 15 minutes. To this end, Specter et al. formalize and instantiate a new cryptographic

[4]One might think that DKIM protection increases email deniability, since anyone with access to the email server—such as an employee—can forge signatures. In reality, DKIM signing keys are kept private by email providers, as losing these keys would enable internal or external adversaries to mount email-based attacks such as phishing.

primitive called forward forgeable signatures. One instantiation of such signatures is KeyForge, which builds on a so-called hierarchical signature scheme. This approach preserves the protection of DKIM during the delay, but enables anyone to forge DKIM signatures after the delay, thereby achieving a form of deniability. However, after the delay, unless Alice's account is compromised, Bob cannot spoof Alice's identity to authenticate to the server, precluding deniability.

To address this, the authors of KeyForge propose KeyForge$^+$ [SPG21], which includes an additional protocol called forge-on-request. This protocol enables parties like Bob to request forged emails from any domain, such as Alice's domain, under the only condition that the recipient of the forged email is the requester, i.e., Bob in this case. In other words, Bob can request to forge an email from Alice only if the recipient is Bob himself. For group emails, Bob can request a forged email from any domain. This forge-on-request protocol enables the simulator to spoof Alice's identity and authenticate to the server.

Model variables. To capture plain DKIM [CHK11], KeyForge and KeyForge$^+$ [SPG21], we can choose model variables as follows:

- pk_A: This corresponds to Alice's domain's public key.
- sk_B: Bob's domain's secret key or $\perp$, depending on the power of the judge, albeit it should not affect its decision.
- $f(\mathsf{st}_S)$: This depends on context - see the discussion below for how server leakage can prevent deniability.
- st'_B, aux and the judge: These are context-dependent as before. In the case of the Clinton e-mail leaks, the judge was the general public and the press (note the similarity between this scenario and Example 61).
- Oracle $\mathcal{O}$: For plain DKIM, the oracle behaves like AUTH, given that the signing key is never leaked. For KeyForge for a given DKIM signing key, the oracle behaves as in AUTH until the signing key is released, after which it behaves like $\mathsf{NO\text{-}AUTH}$. For KeyForge$^+$, when the signing key is not released, it behaves like $\mathsf{NO\text{-}AUTH}$ if a request is made with forge-on-request (note we assume parties honestly execute the protocol, so such a request should succeed), and otherwise as in KeyForge.

In our model

Forging DKIM signatures is not enough to achieve practical deniability. For Alice, the victim, to defend herself against Bob, the accuser, Bob must be able to send an email on Alice's behalf, which is not immediately possible in KeyForge, even if Alice's email is registered on a different domain. This limitation is also visible in the AUTH oracle: on line 5, the oracle uses Bob's simulated state ($\widetilde{\mathsf{st}_B}$). This is

resolved in KeyForge$^+$ as described above. That is, until the signing key is public, the oracle cannot forge a DKIM signature and therefore Bob's state is not updated (the Vf function on line 2 of AUTH in Figure 2.2 returns false). When the signing key becomes publicly available, anyone can forge DKIM signatures: the Vf function returns true and the oracle can update Bob's state.
The combination of KeyForge and KeyForge$^+$ shifts the oracle from AUTH to NO-AUTH after a predefined delay, thereby increasing the deniability properties of DKIM-protected email. However, our model demonstrates that the server plays a crucial role in real-world deniability. The judge inputs a function of the real and simulated server states (the former in the real world, the latter in the ideal one), where the function f models different leakages from the server. The work of Specter et al. does not account for this leakage: the server could log Bob's request of a forgery of Alice's messages differently than how it logs a real sending procedure from Alice, which would help the judge to distinguish the real and ideal worlds. Additionally, the server could use two different databases to store forgeries and legitimate emails. The judge can subpoena these databases to differentiate between the two worlds [Cas20]. This highlights the significant role that the service's server (formally represented by the function f) plays in our model.

Summary and discussion

In the above discussion we saw that plausible deniability for DKIM-protected email is non-existent and how KeyForge and KeyForge$^+$ partially solve this problem. At a high level, the procedure is similar to what we propose for Signal: use "cracking" as a controlled and desirable feature. In this case this involves releasing the private signing keys after a predefined amount of time, which breaks the authentication that DKIM provides. However, as with Signal, email servers must play the game: if they log too much information about the exchanged messages and the judge obtains this information, deniability is lost.

Unlike our proposed approach for Signal, i.e., giving the ability to edit messages on the user interface level to all users, KeyForge$^+$ requires important changes to the current email infrastructure in order to be deployed, as acknowledged by the authors. This forge-on-request protocol must also be easy to use and available to everyone to avoid the unequal access to deniability that arises when forging is accessible only to tech-savvy users. Additional research is thus necessary to assess the usability of KeyForge and KeyForge$^+$.

4.5 Legal case studies

This section considers deniability in the legal context by presenting our analysis of court cases that mention messaging applications in Switzerland. To the best of our knowledge, this is the first such analysis capturing the civil law paradigm (the studies that we cite in Section 4.2.2 considered participants or cases in the United States) and therefore fills a gap that Yadav et al. highlighted [YGS23].

Yadav et al. conducted an analysis of 228 court cases in which WhatsApp conversations were proposed as evidence. They found that WhatsApp chats where *never* rejected as evidence because of their cryptographic deniability properties, even when the defendants claim that conversations can be forged. One illuminating example from their analysis is the United States v. Ojimbda case:

> Defendant's objection that the text messages, in this case, are unreliable is made *without any persuasive evidence* and is thus overruled... The court explained that Mr. Ojimba could attack the reliability of the messages at trial, but that reliability was ultimately a matter for the jury.

We conducted a similar analysis in Switzerland. We start by introducing our methodology and focus later on the results.

4.5.1 Methodology

Our legal analysis aims at answering three research questions. We settled for WhatsApp (which is more widely used) in our research as none of the decisions we surveyed mentioned Signal. Even if WhatsApp is less deniable than Signal because it is less private, i.e., additional metadata may be available to an adversary, we believe that a court's opinion would be similar or the same for Signal or other messaging applications. This is especially true if the court does not subpoena the relevant service provider for metadata, which is true for the Swiss cases we considered. The three research questions are as follows:

1. Do judges in Swiss courts use WhatsApp messages as evidence?
2. When WhatsApp messages are used as evidence, is their usage contested by any of the parties involved?
3. What are the specific reasons cited for disputing the legal validity of WhatsApp messages, and how do judges respond to these disputes?

Our hypothesis was twofold: Firstly, judges in Swiss courts use WhatsApp messages as evidence in penal cases. Secondly, the validity of these messages is generally not

disputed by the parties involved, despite the cryptographic deniability properties inherent in WhatsApp's algorithm, i.e., the same as Signal.

```
"query": { "bool": {
  "must": [
    { "match": {
        "attachment.content": "WhatsApp"
      }
    }
  ],
  "should": [
    { "match_phrase": {
        "attachment.content": "droit pénal"
      }
    },
    { "match_phrase": {
        "attachment.content": "chambre pénale"
      }
    },
    { "match_phrase": {
        "attachment.content": "diritto penale"
      }
    }
  ],
  "minimum_should_match": 1
}}
```

Figure 4.4: Elasticsearch query for the legal analysis.

To design our study, we consulted with four Swiss lawyers. We used the website https://entscheidsuche.ch, a private initiative that publishes decisions from all instances of Swiss courts. We queried the website with an Elastic search query (Figure 4.4) to select the initial cases, i.e., penal cases in French or Italian that contain the word "WhatsApp". We opted for cases in French and Italian because we speak these languages and did not use any translation tool. We focused on penal cases to analyze those with a significant coercive impact on the defendants. As of February 22, 2024, our query returned an initial set of 419 decisions from all instances. From this initial set we removed all the decisions from federal courts by removing the corresponding URLs from the query results (e.g., `CH_BGer` for supreme court). We opted for this additional pruning for two reasons:

1. The federal criminal court treats cases originating from the office of the attorney general of Switzerland, involving major investigations where conversations play a minimal role, if any.

2. The federal supreme court, the highest court in Switzerland, has jurisdiction over different violations, including penal cases. However, this court does not accept the

addition or removal of evidence: the set of available evidence is fixed at the first two levels of the Swiss legal system.

We thus focused on the first two level of Swiss courts with jurisdiction over penal cases, leaving us with a set of 341 decisions. We performed an additional verification of decisions from the supreme court, as it is the court most likely to rule against accepting messaging conversations as evidence if such a decision is appealed. We found no such ruling, and the lawyers confirmed that it does not exist.

We manually analyzed the 341 cases and divided them into four categories. When in doubt, we consulted with the four lawyers who helped us design the analysis. The four categories are as follows.

- N/A: The decision mentions the word "WhatsApp" for purposes other than evidence, such as to summarize police investigations or to mention that two parties can or cannot contact each other via WhatsApp. We also put outliers in this category (e.g., administrative cases that the website wrongly returns as penal ones or federal courts cases that we missed during the initial pruning).
- Evidence: WhatsApp chat is used as an evidence in the final decision.
- Contested: One of the parties contests the validity of WhatsApp conversation as legal evidence.
- Rejected: The judges decide to reject WhatsApp chat as evidence after a party contested its validity.

In the full version of the extended abstract corresponding to the work in this chapter we provide the URLs for the 341 cases (from the website https://entscheidsuche.ch) [CCHD23, Appendix B.2]

4.5.2 Results

Total Cases	N/A	Evidence	Contested	Rejected
341	201 (59%)	140 (41%)	2	0

Table 4.1: Summary of legal case analysis results.

Table 4.1 summarizes the results of our analysis. We find that 59% of the cases that mention the word "WhatsApp" do not report the use of conversations as evidence. Conversely, for 41% of the decisions that we analyze, WhatsApp conversations are used as evidence. In all except one case that we discuss below, the authenticity of messages,

or their legal validity, is never questioned by any of the parties involved. In particular, deniability is never invoked (successfully or otherwise) in all the 140 cases in which electronic messages are used as evidence. We find *two* cases in which a party in an appeal denies being the author of messages appearing on a screenshot. In the first decision (Arrêt en ligne de la Chambre pénale du Tribunal cantonal du canton de Fribourg 502 2020 76 du 11 août 2020) we can read (text translated by the authors):

> On the other hand, she disputes having written or sent the WhatsApp messages contained in the Justice of the Peace's file, even though they were produced by C._____ on November 21, 2017. She asserts that the documents could have been altered or that there might have been manipulation on the WhatsApp application regarding the name of the sender of the messages.[5]

Despite the dispute, the judges decided not to reject the evidence or pursue the matter further (for example by ordering a forensic analysis of the appellant's phone). The judges deem the appellant's argumentation not credible and they posit that their invalidation would not change the original ruling:

> Her argument is far from convincing. Under these circumstances, there is no reason to order an analysis of the parties' mobile phones. Moreover, even if such an analysis were ordered and it demonstrated that the messages did not originate from the appellant, this would still not constitute sufficient suspicion that the respondent has committed one or more criminal offenses [...].[6]

In the second decision (Arrêt en ligne de la Chambre pénale d'appel et de révision de la Cour pénale de la Cour de justice du canton de Genève AARP/54/2018 du 22 février 2018), an appellant claims that "he was not the author of the messages that B_____ had received [...]"[7]. This claim builds on the idea that someone sent messages to the potential victim posing as the plaintiff, using a new phone number. As in the first case, the judges considered the arguments not credible. In particular, the content of the conversation and the "Machiavellian sense" to mount such an attack convinced the judges that the author of the messages can only be the appellant.

[5] Original text: "D'autre part, elle conteste avoir écrit ou envoyé les messages WhatsApp contenus dans le dossier de la Justice de paix, quand bien même ceux-ci ont été produits par C._____ le 21 novembre 2017. Elle soutient qu'il pourrait s'agir soit de documents qui ont été modifiés, soit d'une manipulation effectuée sur l'application WhatsApp quant au nom de l'expéditeur des messages."

[6] Original text: "Son argumentation ne convainc ainsi pas, loin s'en faut. Dans ces circonstances, il n'y pas lieu d'ordonner une analyse des téléphones portables des parties. Par ailleurs, même dans l'hypothèse où une telle analyse était ordonnée et qu'elle démontrait que les messages n'émanaient pas de la recourante, cela ne constituerait pas encore un soupçon suffisant que l'intimé s'est rendu coupable d'une ou plusieurs infractions pénales [...]".

[7] Original text: "Il n'était pas l'auteur des messages que B_____ avait reçus [...]".

Additional results

Our analysis enabled us to find other interesting results besides the primary focus that our research questions set; here we present some of them.

WhatsApp as an investigative tool. From one of the cases that we analyze (Arrêt en ligne de la Chambre pénale du Tribunal cantonal du canton de Fribourg 502 2022 8 du 24 janvier 2022), we evince that an undercover policeman used WhatsApp to communicate with a potential pedophile.

> [...] From March 10 to 29, 2021, he maintained a virtual exchange via e-mails and then WhatsApp messages with a young boy [...] who had indicated that he was 14 years old. On March 29, 2021, the appellant invited the child to a hotel [...] pretending to be his nephew, in order to have intimate relations [...]. In reality, he was conversing with a police officer.[8]

Disappearing messages as additional evidence. The "disappearing messages" feature, available in most modern messaging systems, enables users to send messages that automatically delete after a set period. We found some decisions in which the use of this feature is highlighted by the judges. In one case in particular (Arrêt en ligne de la Chambre pénale d'appel et de révision de la Cour pénale de la Cour de justice du canton de Genève AARP/309/2022 du 10 octobre 2022), disappearing messages represent additional incriminating evidence:

> Finally, the fact that I_____ and the appellant were initially represented by the same lawyer, on the one hand, and that their WhatsApp exchanges have disappeared from the appellant's phone, on the other hand, are additional incriminating indicators.[9]

Screenshots as evidence. In most of the cases that we analyze, the decision does not specify how parties present messages to the court. In some of them, the judges specify that parties present evidence through screenshots, without discussing or analyzing their truthfulness or chain of custody. Our analysis does not evidence any discussion of potential tampering or forging of these screenshots, for WhatsApp conversations or other evidence.

[8] Original text: "[...] Du 10 au 29 mars 2021, il a entretenu un échange virtuel via courriers électroniques puis par messages WhatsApp avec un jeune garçon [...] qui lui avait indiqué avoir 14 ans. Le recourant a invité l'enfant à se rendre le 29 mars 2021 dans un hôtel [...] en se faisant passer pour son neveu pour y entretenir des relations intimes [...]. En réalité, il conversait avec un policier."

[9] Original text: "Enfin, le fait que I_____ et l'appelant étaient initialement représentés par la même avocate, d'une part, et que leurs échanges Whatsapp ont disparu du téléphone de l'appelant, d'autre part, sont des indices de plus à charge."

4.5.3 Analysis

The results we present in the previous section confirm our first hypothesis: judges use chats as evidence in penal cases. The validity of these messages is generally not disputed by the parties involved despite cryptographic deniability properties. Our study, along with Yadav et al.'s findings [YGS23] show that deniability seems irrelevant in court cases in Switzerland and the United States. Although the results cannot be generalised to other countries, they provide evidence that cryptographic deniability does not translate into real-world deniability and that a purely cryptographic approach does not impact the legal setting. When one party claims a forgery, judges consistently reject these claims. In what follows, we analyze possible reasons for this failure discuss potential countermeasures.

Lawyers, judges, and authorities are generally unaware of cryptographic deniability and its workings. The use of WhatsApp by the police as an investigative tool highlights this lack of knowledge. This results from society's lack of acceptance of deniability. Therefore, if deniability is a desired feature, it must be socially accepted, simple and practical for everyone. Simplicity could also limit the imbalance in access to justice when deniability could be a factor. Malicious parties could exploit deniability to forge messages that are considered real by courts, while wealthy defendants could hire experts to testify about deniability properties, an option not available to most people dealing with low-profile cases.

The two cases that we present in Section 4.5.2 highlight the importance of contextual information for deniability, in court or other settings. Communication transcripts are (1) one type of evidence among others with varying degree of importance (as in the first case discussed), and (2) can be authenticated given other contextual information (as in the second case). The first aspect is important: even with complete real-world deniability, judges would have the ability to convict perpetrators of particularly heinous crimes, several examples of which we encountered in our analysis.

In our model

We represent contextual information with auxiliary data in our model. Auxiliary data enables the judge to input different types of evidence with varying degree of importance. We assume the judge inputs all evidence through auxiliary data and then decides their weight in the specific case. The value ν, which models the *in dubio pro reo* principle, captures the discretion that judges are afforded in evaluating each case.

In several cases, screenshots are accepted as evidence. This finding has several implications. Despite the prevalence of graphics editors, synthetic media such as deepfakes, and misinformation [CN21], people tend to believe that a simple screenshot can be accepted as evidence—as confirmed by the two survey studies detailed in Section 4.2. This implies once again a risk of unbalanced access to the deniability property: a malicious individual

could use deniability to produce fake evidence against a victim and judges would likely consider this evidence as acceptable given our findings.

Finally, during our analysis we observed that judges believe that the law protects against forgeries, as false testimony is a criminal offense. This means any transcript, even a forgeable one like a screenshot, can be considered in court. This highlights the need for a discussion about deniability in various settings, as partially addressed by Yadav et al. [YGS23]. We expand on this in the discussion below.

4.6 Discussion

In the previous sections we show how (cryptographic) deniability fails in practice from both a technical perspective—through the lens of our model defined in Section 4.3—as well as a legal perspective. This section elaborates on these findings on practical deniability and their consequences.

4.6.1 Either practical deniability or no deniability

Cryptographic deniability does not translate in the real world and appears to not be used in practice, which we have justified in the previous sections. The first question is therefore whether deniability is a necessary or desirable feature. Obtaining real deniability in practice implies non-trivial challenges that can impact users (e.g., spamming), performance (e.g., the cost of adding deniability in the protocol, attacks on sealed-sender-based authentication [TLMR22]), the service (e.g., respectability) or security (e.g., authenticity is somewhat lost if Bob can easily forge messages from Alice). Indeed, Apple has decided not to target deniability in PQ3, their new messaging protocol for iMessage [Ste24].

Besides the technical challenges, it is important to consider whether deniability is a *beneficial* property. Practical deniability could facilitate the spread of "fake news" by enabling everyone to forge messages. Also, in the jurisprudence we reviewed, messages on victims' phones are used to convict criminals: practical deniability could invalidate this kind of evidence. Deniability would for example prevent a victim that receives abusive messages from proving the culpability of their author. However, as our legal analysis shows, conversations are usually one piece of evidence among others and do not represent the decisions' cornerstone. We also recall that the study of Yadav et al. [YGS23] found that 60.2% of users surveyed desire only non-repudiation (i.e., they do not desire deniability), while only 12.7% desire deniability.

The very nature of deniability makes it hard to settle for either real-world deniability or no deniability at all. Differently from other security properties (e.g., encryption), users cannot control the deniability of their communication: deniability is effective only if *receivers* can plausibly forge messages. This however should not serve as an excuse to not consider

vulnerable populations and to generalize survey results to different contexts [YGS23, Section 8]. Unfortunately our field lacks research on practical defensive technologies that vulnerable groups use. One important exception is the work by Daffalla et al. [DSKB21] on the technological defense strategies that political activists used during the Sudanese revolution, which indicates that deniability is useful in this setting. Some participants used strategies to increase plausible deniability in case of arrest, such as adding decoy messages on their messaging application. Having complete deniability, such as ability to edit the user interface of Signal as shown in Figure 4.1, would therefore help in cases where there is a need for a way to communicate freely without being held accountable, as the interviews of Daffalla et al. confirm.

If we opt for deniability, this should be *practical and accessible to everyone*. As we highlight in previous sections, impractical and unusable deniability is dangerous as only a few people can benefit from it. This was also pointed out by Jeff Burdges[10]: if deniability is not practical and widespread, the risk is that only rich and powerful people would have the resources to argue deniability in a court of law (or against the public opinion).

This discussion is reminiscent of the debate on end-to-end encryption, but the impact of deniability is less clear. Encryption solves, among other things, the problem of mass surveillance: an attacker has no access to messages in transit. Encryption is a mathematically-based property that has an immediate and global effect. By contrast, deniability is a more subtle, subjective property, which involves human and societal factors that can vary from case to case, e.g., which auxiliary information the judge has or which contextual information can help to frame the victim. Additionally, deniability holds if the *receiver* of a message satisfies some requirements: the author of a message cannot completely control which kind of deniability a message gets, whereas the sender, i.e., the author, has complete control over the message's encryption. Since deniability is not used in practice, it is difficult to argue about its impact and a community-wide discussion is necessary.

Finally, one might wonder if the strong deniability property that we seek is actually anonymity. By making it difficult for the judge to identify, profile and obtain data about the sender, or to link an account to them, we might avoid the deniability question in many cases. Achieving real-world plausible deniability in public messaging applications like WhatsApp and Signal, which link accounts to identities (e.g., phone numbers) seems very challenging. While allowing users to edit or add messages to a conversation would be a step towards this goal, it is unclear whether society would accept this as true deniability.

[10] https://mailarchive.ietf.org/arch/msg/mls/Kk6qai3kEza8B-L-_5R-qsEjtK0/

4.6.2 Deniability in front of whom

If real-world deniability is a goal for a given system, one must devise an appropriate threat model. The design must precisely define the judge in Definition 57. Who are we defending against? Which capacity and auxiliary information does the judge have? What about the incriminating adversary (Bob in our model)?

The leak of Hillary Clinton's email database [Wik16b] is often cited as an example where deniability would have had an impact. Wikileaks published *DKIM-protected* emails and some of the authors claimed that they were tampered with. However, the DKIM signatures mathematically proved that the leaks were legitimate [Wik16a]: the emails were *not* deniable. Moreover, the legitimacy of the emails is verifiable by the general public: it is sufficient to have the verification keys of the domain servers, publicly available by definition, to prove that no one tampered with the emails' content. Here the *general public* acts as the judge, rather than an official judge in a courtroom as usually envisioned. Our model captures this setting by reflecting real-life scenarios where Bob's data, such as keys and transcripts, are stolen published online, and Alice wants to deny sending the published messages to Bob. We model this by providing the judge with Bob's state st'_B after the interaction with Alice took place and the simulated state $\widetilde{\mathsf{st}}_B$ in the ideal world. This example highlights the importance of (1) properly defining the role of the judge and (2) applying our model to ensure that the judge's distinguishing probability is bounded by an acceptable probability.

As already discussed, if the threat model assumes a global adversary that *actively* colludes with Bob to frame a user, then deniability is impossible to achieve also because a lot of messaging applications register the users' identity (e.g., phone number). In this case, other properties and services such as anonymous communication over Tor should be used: the best way to avoid having to consider deniability in the first place is to guarantee anonymity.

4.6.3 How to make a deniable system

Privacy-focused messaging services like Signal (and others using the Signal protocol) use protocols that achieve a form of cryptographic deniability, like X3DH [MP16] and the recent PQXDH [KS23, FJ24]. These services market deniability as a feature, giving users the impression it is a practical aspect of the system. As we show in Section 4.4.1 the situation is not so simple: Signal in general does not achieve practical deniability, either technically or legally.

To achieve practical deniability, applications must be designed with deniability incorporated by design, similar to privacy. As we hinted before, the system must provide *practical* deniability against the kind of adversaries that the threat model considers. In particular, deniability must be plausible for all and accessible to everyone, regardless of

the (technical) knowledge of the receiver, i.e., the potential accuser Bob in our model. The simulator in our model must also be practical: it should be feasible to "plausibly" run it to modify Bob's state (e.g., the phone).

One way to achieve this is to enable cracking as a controlled feature, allowing users to inject false messages authored by someone else into their conversation. This can be done by offering a feature that enables users to edit their conversation. This modification must be easily accessible in the application's user interface and must also alter messages that the application stores in the local database. The application must not record when and where the modifications take place. This solution mimics the real world: someone may report the content of a conversation, but the veracity of what is reported depends solely on the person reporting it, and other participant can challenge what is claimed. In practice, the modifications to Signal's user interface can mimic the tool used by Reitinger et al. [RMA+23] in their study (Figure 4.1): incoming and outgoing messages can be edited and new messages can be inserted. If the application is offline, editing must not contradict the last access timestamp that the Signal server stores; in case of inconsistency, the application should warn the user. Reitinger et al.'s findings (Section 4.2.2) support this simple solution. In particular, the study suggests than an edit/create message feature significantly impacts people's opinion on someone's guilt, i.e., on plausibility of deniability claims.

Auxiliary data poses a significant threat to deniability. To enhance practical deniability, the system must minimize data retention, keeping only essential information and deleting surplus data once its purpose is fulfilled. This aligns privacy by design principles, which generally increase the plausibility of deniability.

4.7 Conclusion

This work discusses real-world deniability in messaging, highlighting how cryptographic deniability does not translate in the real world, both technically and legally. We analyze the technical side of deniability by applying our model on Signal and email with DKIM protection. In the legal setting, we examine court cases in Switzerland and find no successful claims of deniability in 140 cases where conversations were used as evidence.

We then propose a general discussion about deniability, considering whether it is a property messaging solutions should aim for. If the answer is affirmative, we discuss different settings for real world deniability—in particular who we want to defend against—and how we should design real-world deniable systems. We conclude with some remarks on the technical costs of real world deniability.

Messaging applications, and communication solutions in general, should either provide practical real-world deniability or stop claiming to do so. For messaging applications,

particularly Signal, we propose a simple yet powerful solution: users must be able to modify sent and received messages stored on the device through the user interface. This mirrors the real world, where conversation can be truthfully or falsely reported, and others can challenge those reports. We believe Signal should adopt this approach to offer truly deniable communication, especially given its strong privacy focus.

5 Conclusion

> At this point, I think we would do well to put ourselves in the mindset of a *real* adversary, not a notional one: the well-funded intelligence agency, the profit-obsessed multinational, the drug cartel. You have an enormous budget. You control lots of infrastructure. You have teams of attorneys more than willing to interpret the law creatively. You have a huge portfolio of zero-days. You have a mountain of self-righteous conviction. Your aim is to *Collect it All, Exploit it All, Know it All.*
>
> Phillip Rogaway, *The Moral Character of Cryptographic Work* [Rog15, Part 4]

This thesis addresses three practical challenges for secure messaging: active attack detection (Chapter 2), private and secure public key retrieval in (Chapter 3) and deniability (Chapter 4). These topics reflect critical aspects of secure messaging, and communication, systems, from ensuring resilience against active adversarial attacks to safeguarding metadata privacy and providing plausible real-world deniability for users. In this concluding chapter, we summarize the key contributions of the thesis and explore potential directions for future research. This includes a high-level discussion of how the three contributions integrate to enhance the overall security and privacy of messaging systems, as well as specific avenues for improving the individual solutions presented in each chapter.

5.1 Summary

In Chapter 2, we propose cryptographic protocols to detect active attacks in messaging schemes that support immediate decryption. We present protocols that use the in-band channel (i.e., the same channel that parties use to send messages) and protocols that rely on an always authentic out-of-band channel. For both settings we introduce two uncomparable notions: r-RID and s-RID for the in-band setting, and r-UNF and s-UNF for the out-of-band setting. Together, these security notions ensures RID and UNF respectively. We propose schemes that are secure under these notions, making only black-box use of a classic ratcheted communication scheme. The chapter concludes with a discussion on the practicality of these notions, and optimizations and security/performance trade-offs, paving the way for practical active attack detection in messaging schemes with immediate decryption, such as Signal's Double Ratchet [PM16].

In Chapter 3, we shift to a higher level of abstraction and address metadata leakage in messaging systems. We introduce authenticated PIR, a new cryptographic primitive that guarantees both privacy and integrity of the retrieved data. We define authenticated PIR in both multi-server and single-server settings and propose schemes for each. The multi-server scheme for point queries (i.e., privately fetching a record from the database) combines a classic PIR scheme with a Merkle tree. The multi-server scheme for predicate queries (i.e., privately evaluating a function on the database records) uses correlated queries with function secret sharing to achieve both privacy and integrity. All schemes are secure against selective-failure attacks. We evaluate all schemes and use the multi-server ones to design our motivating application Keyd, a privacy-preserving PGP key directory server.

Chapter 4 examines how cryptographic deniability interacts with real-world conditions. We move to an even higher level of abstraction, analyzing how cryptographic solutions, applications, and the devices on which they run ensure (or fail to ensure) the properties expected from modern secure messaging systems. To this end, we propose a new model for real-world deniability and apply it to both the Signal application and DKIM-protected email. We demonstrate that these systems do not offer practical deniability guarantees. Additionally, we analyze 140 court cases in Switzerland that use messaging conversations as evidence, finding no claims of deniability, which suggests that this property has no practical impact in legal contexts. Based on this analysis, we discuss whether deniability is a desirable property and, if so, how to design systems that are deniable in practice. For Signal, we propose a straightforward yet effective solution: the application should allow users to directly modify locally stored messages via the user interface.

5.2 Future work

In this section we discuss some possible venues for future work. We start by a general discussion on the future work that can build in the *entirety* of this thesis and then we present some specific venues for each of the three chapters that compose this manuscript.

Real adversaries. We began this thesis with a quote from Philip Rogaway's essay *The Moral Character of Cryptographic Work* [Rog15] and we conclude with another quote from the same essay that urges the cryptographic community to consider *real* adversaries. Throughout this thesis, we have followed Rogaway's advice by addressing malicious adversaries across various levels of the secure messaging stack. In Chapter 2, we consider an adversary that can impersonate parties, either temporarily or indefinitely, expose secret keys and states, and manipulate the randomness used by cryptographic algorithms. Chapter 3 addresses a potentially malicious messaging operator, we address a potentially malicious messaging operator that can tamper with public encryption keys during the key distribution process, posing a security threat, while also collecting sensitive metadata about communication patterns, presenting a privacy risk. Moreover, we consider selective-failure attacks that such malicious PKI operators can mount. Finally, in Chapter 4, we take this a step further by considering adversaries with access to user's devices and other components of the messaging system, such as the server responsible for client authentication and message forwarding. Future work, particularly in secure messaging and more broadly in information security research, must continue to account for these powerful, real-world adversaries in the design and analysis of privacy and security solutions. Concretely, we propose two directions for future research, although one has already been explored by Dietz and Tessaro, who at CRYPTO 2024 [DT24] introduced a single-server authenticated PIR scheme based on the decisional Diffie-Hellman assumption, tolerating a fully malicious server that provides a potentially incorrect digest of the database. As Dietz and Tessaro point out, designing such a system under the LWE assumption remains an open problem. Another direction pertains to real-world deniability: future work could examine potential adversaries attempting to detect modifications to the local message database to compromise deniability. This includes considering side channels and residual data, such as deleted data lingering in system I/O buffers or temporary storage, which might persist on a device after a conversation.

Group messaging. Chapter 2 and Chapter 4 focus on the two-party setting in messaging, where our protocols and analysis assume a conversation between Alice and Bob only. An interesting direction for future work would be to extend these concepts to the group setting, where more than two parties are involved in the conversation. For instance, a promising direction for future work is to define RID and UNF security notions and corresponding constructions for group messaging. Similarly, it would be valuable to extend our model for real-world deniability and corresponding analysis to the group setting, exploring cases where, for example, judge corroborates with—or a party tries to

frame—multiple group members.

Concrete performance. Concrete performance is crucial to enable adoption of solutions that strength the security and privacy of messaging solutions. In this thesis we thoroughly evaluate our authenticated PIR schemes and apply the multi-server ones to Keyd, a privacy-preserving PGP key directory server. However, our single-server schemes are 30-100× more costly than state-of-the-art unauthenticated single-server schemes, even if they achieve incomparably stronger integrity properties. One interesting venue for future work is to construct single-server authenticated-PIR schemes whose performance matches that of the best unauthenticated schemes. Also the work on active attack detection that we present in Chapter 2 would strongly benefit of an implementation and evaluation of the schemes that we present, with the goal of analyzing the practical overhead of in-band and out-of-band authentication.

User studies and multidisciplinarity. Information security in general, is not solely the result of mathematical and computer science solutions but is shaped by a complex relationship with the real world. It is essential to design and evaluate secure messaging solutions by considering who uses them, who the real adversaries are, how the political and societal context influences security and privacy, and how the legal framework may (or may not) account for these properties. To this end, engaging with the end-users and collaborating with other disciplines is crucial for understanding the real-world impact of our solutions. Concretely, more user studies on practical deniability are needed. The studies [RMA+23, YGS23] that we discuss in Chapter 4 do not involve real messaging applications, so it would be valuable to observe how users react when conversations can be modified within the interface. While writing this thesis, Rajendran et al. [RYAJ+24] published a paper that presents a study in which users interact with the transcript-editing feature that we propose for Signal. The authors highlight that this hands-on experience offers a novel approach to evaluating the usability of the solution we propose for real-world deniability in Signal. Additionally, future research could explore whether deniability is desirable in different contexts, especially for vulnerable groups such as political dissidents, whistleblowers, or harassment victims.

A Supplementary material for Chapter 2

This section presents supplementary material for Chapter 2, On active attack detection in messaging with immediate decryption.

A.1 Proof for Theorem 10

We present the proof for s-RID security of the construction introduced in Figure 2.7.

Proof. The proof strategy is identical to the one done for the proof of Theorem 9. For any adversary $\tilde{\mathcal{A}}$ playing the s-RID game, we construct an adversary $\mathcal{A}^*$ playing the CR game with comparable complexity. We first describe the adversary $\mathcal{A}^*$ in terms of $\tilde{\mathcal{A}}$ and proceed by proving that $\mathcal{A}^*$ wins at least as often as $\tilde{\mathcal{A}}$.

As with the proof of Theorem 9 we define an event E that occurs only when $\text{s-RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1$, and we prove that $\Pr[\text{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 | E] = 1$.

The event $\text{s-RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1$, means there exists $\mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y$ such that both $x < y$, $(\mathsf{num}, \mathsf{pt}, \mathsf{ct})$ is a forged message received by $\overline{\mathcal{P}}$ at time x, $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ is an honest message sent by $\overline{\mathcal{P}}$ at time y and received by $\mathcal{P}$. As $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ was received, $\mathcal{P}$'s call to RRC.checks returned false.

Let us define $h_f = \mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$. When receiving the forged message, i.e., at time x, $\overline{\mathcal{P}}$ adds (num, h_f) to $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$. As $y > x$ at time y, (num, h_f) is still in $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$. Hence $\mathsf{num} \in \mathsf{nums}'$ for the honest message $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$. Now as $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ was accepted, we have, due to line 3 of checks, that

$$\mathsf{H.Eval}(\mathsf{hk}', \mathsf{R}^*) = \mathsf{H.Eval}(\mathsf{hk}', \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}).$$

If $\mathsf{R}^* \neq \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$ we have already found a collision. So let us assume $\mathsf{R}^* = \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$. Now as $(\mathsf{num}, h_f) \in \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$, we also have that $(\mathsf{num}, h_f) \in \mathsf{R}^* \subset \mathsf{st}_{\mathcal{P}}.\mathsf{S}$.

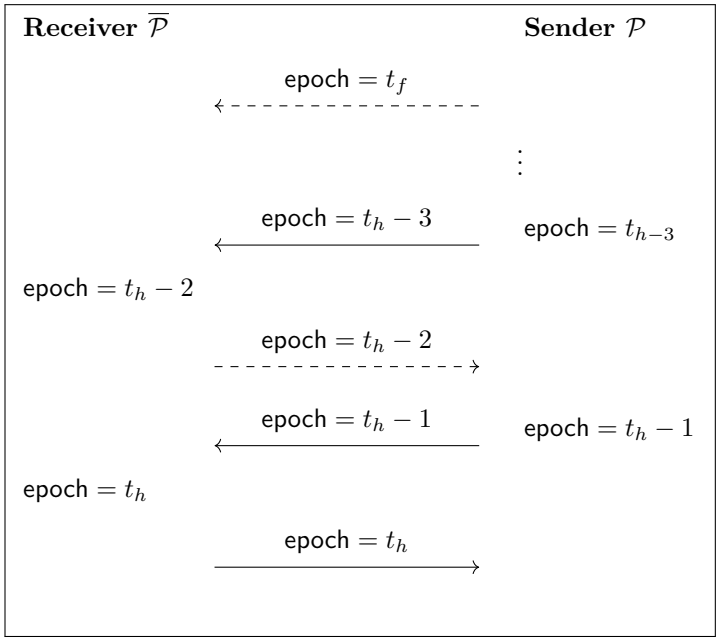

Figure A.1: Visualising the proof sketch of Theorem 15. Dotted lines represent forged messages, while solid lines represent honest messages. Intuitively, we argue that if the last message did not contradict the first, then the fourth message would have contradicted the third, thereby violating s-RID in the other direction.

This would mean that there exists an honest message $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)$ such that

$$\mathsf{H.Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)) = h_f.$$

But note that as $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)$ is an honest message, $(\text{“}\mathsf{send}\text{”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \in \log$ but $(\text{“}\mathsf{send}\text{”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \log$ as the message was forged, hence $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \neq (\mathsf{num}, \mathsf{ad}, \mathsf{ct})$, which again yields a collision pair.

Now the CR adversary $\mathcal{A}^*$ does the following: they run the s-RID adversary $\tilde{\mathcal{A}}$ as a subroutine with the hk given by the CR game. They later find $\mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}\ , \mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y$ satisfying the condition, in case they exist. Now by exposing the states of the parties at time y they can find the collision pairs described above. Hence we have,

$$\Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1] \geq \Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 | E] \cdot \Pr[E] = \Pr[\mathsf{s\text{-}RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1]$$

Moreover the run-time of $\mathcal{A}^*$ is roughly the run-time of $\tilde{\mathcal{A}}$. □

A.2 Proof sketch of Theorem 15

We present the proof sketch for s-RID security of the construction introduced in Figure 2.13.

Proof sketch. Let $[(\mathsf{ct}_f, \cdot, t_f), (\mathsf{ct}_h, \cdot, t_h)]$, be the closest pair of sent-received messages contradicting the s-RID condition. Meaning $(\mathsf{ct}_f, \cdot, t_f)$ is a forgery received by $\overline{\mathcal{P}}$ before they sent the honest message $(\mathsf{ct}_h, \cdot, t_h)$ which was received by $\mathcal{P}$. We consider the time when $(\mathsf{ct}_h, \cdot, t_h)$ was sent by $\overline{\mathcal{P}}$. As mandated by the construction $(\mathsf{ct}_h, \cdot, t_h)$ contained num values and accumulated hash of all messages $\overline{\mathcal{P}}$ has received at epochs t_h and $t_h - 2$. Following the same argument as the proof of Theorem 10 one can show that no forgeries, including $(\mathsf{ct}_f, \cdot, t_f)$, were received by $\overline{\mathcal{P}}$ while $\mathsf{epoch}_{\overline{\mathcal{P}}} \in \{t_h, t_h - 2\}$.

The two messages that changes $\mathsf{epoch}_{\overline{\mathcal{P}}}$ from $t_h - 4$ to $t_h - 2$ and from $t_h - 2$ to t_h, were honest messages by $\mathcal{P}$. Let us call them $(\mathsf{ct}_{t_h-3}, \cdot, t_h - 3)$ and $(\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)$ respectively. Note that, both these messages were received after $(\mathsf{ct}_f, \cdot, t_f)$ was received and before $(\mathsf{ct}_f, \cdot, t_f)$ was sent, as otherwise $(\mathsf{ct}_h, \cdot, t_h)$ would contradict $(\mathsf{ct}_f, \cdot, t_f)$. Note that between sending the messages $(\mathsf{ct}_{t_h-3}, \cdot, t_h - 3)$ and $(\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)$, $\mathsf{epoch}_{\mathcal{P}}$ was changed meaning $\mathcal{P}$ received a message with $\mathsf{epoch} = t_h - 2$ that caused the change of $\mathsf{epoch}_{\mathcal{P}}$. Let us call this message $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$. We argue this message should have been forged.

Let us assume by contradiction that this message was honest. Note that $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ was sent after $(\mathsf{ct}_{t_h-3}, \cdot, t_h - 3)$ was received, hence after $(\mathsf{ct}_f, \cdot, t_f)$ was received. Now if $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ is honest it forms a pair with $(\mathsf{ct}_f, \cdot, t_f)$ which violates the s-RID condition and has less distance from the original pair which is a contradiction. So $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ must have been forged. Figure A.1 provides a visualization of this scenario.

One other observation is that $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ was received before $(\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)$ was sent, hence, before $(\mathsf{ct}_h, \cdot, t_h)$ was received. This shows that the pair $[(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2), (\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)]$ also violates the s-RID (for $\overline{\mathcal{P}}$ and not $\mathcal{P}$) and has less distance than the original pair. □

B Supplementary material for Chapter 3

This section presents supplementary material for Chapter 3, Authenticated private information retrieval.

B.1 Amplifying integrity in single-server authenticated PIR

In this section, we formally describe how to amplify integrity in single-server authenticated PIR (Section 3.3.3). We begin by defining error-correcting codes, followed by presenting a formal single-server construction for amplifying integrity (Construction 5) along with the corresponding security proofs.

Definition 62 (Error-correcting code)**.** A (k, n)-error-correcting code over a finite field $\mathbb{F}$ that can correct up to t errors consists of two efficient and deterministic algorithms:

- $\mathsf{Encode}(\mathbf{x}) \to \mathbf{y}$: The encoding algorithm takes a message $\mathbf{x} \in \mathbb{F}^k$ and outputs a codeword $\mathbf{y} \in \mathbb{F}^n$.
- $\mathsf{Decode}(\mathbf{y}) \to \mathbf{x}$: The decoding algorithm takes a codeword $\mathbb{F}^n$ and outputs a message $\mathbf{x} \in \mathbb{F}^n$.

Moreover, for all $\mathbf{x} \in \mathbb{F}^k$, $\mathbf{y} \leftarrow \mathsf{Encode}(\mathbf{x})$, and all $\mathbf{y}' \in \mathbb{F}^n$ such that $\mathbf{y}_i = \mathbf{y}'_i$ for all but at most t indices $i \in [n]$, $\mathsf{Decode}(\mathbf{y}') = \mathbf{x}$.

Construction 5 shows how to use an error-correcting code to amplify the integrity of an authenticated single-server PIR scheme, which can have a non-negligible integrity error. Correctness of Construction 5 follows by construction. Thus, we focus on analyzing integrity and privacy.

Construction 5 (Amplifying integrity of single-server authenticated PIR)**.** Let $\mathsf{ECC} = (\mathsf{Encode}, \mathsf{Decode})$ be a (k, n)-error-correcting code over a finite field $\mathbb{F}$ that can correct up to t errors. Let $\mathsf{PIR}_0 = (\mathsf{Digest}_0, \mathsf{Query}_0, \mathsf{Answer}_0, \mathsf{Reconstruct}_0)$ be a secure single-server authenticated PIR scheme for records in $\mathbb{F}$ and which provides ϵ-integrity. We construct a new single-server authenticated PIR scheme from PIR_0 with records in $\mathbb{F}^k$.

$\underline{\mathsf{Digest}(1^\lambda, \mathbf{x} \in (\mathbb{F}^k)^N) \to d}$

1. Parse $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ where $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{F}^k$.
2. For each $i \in [N]$, let $\mathbf{y}_i \leftarrow \mathsf{Encode}(\mathbf{x}_i) \in \mathbb{F}^n$. Write $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,n})$.
3. For each $j \in [n]$, let $\mathbf{z}_j = (y_{1,j}, \ldots, y_{n,j}) \in \mathbb{F}^N$.
4. For each $j \in [n]$, compute $d_j \leftarrow \mathsf{Digest}_0(1^\lambda, \mathbf{z}_j)$ and output $d = (d_1, \ldots, d_n)$.

$\underline{\mathsf{Query}\,(d, i \in [N]) \to (\mathsf{st}, q)}$

1. For each $j \in [n]$, sample $(\mathsf{st}_j, q_j) \leftarrow \mathsf{Query}_0(d_j, i)$.
2. Output $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_n), q = (q_1, \ldots, q_n)$.

$\underline{\mathsf{Answer}\left(d, \mathbf{x} \in (\mathbb{F}^k)^N, q\right) \to a}$

1. Parse $d = (d_1, \ldots, d_n)$ and $q = (q_1, \ldots, q_n)$.
2. For each $j \in [n]$, compute $\mathbf{z}_j \in \mathbb{F}^N$ from $\mathbf{x}$ using the same procedure as Digest.
3. For each $j \in [n]$, compute $a_j \leftarrow \mathsf{Answer}_0(d_j, \mathbf{z}_j, q_j)$ and output $a = (a_1, \ldots, a_n)$.

$\underline{\mathsf{Reconstruct}\,(\mathsf{st}, a) \to \mathbb{F}^k \cup \{\perp\}}$

1. Parse the state $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_n)$ and the responses $a = (a_1, \ldots, a_n)$.
2. For each $j \in [n]$, compute $y_j \leftarrow \mathsf{Reconstruct}_0(\mathsf{st}_j, a_j)$.
3. If there exists $j \in [n]$ such that $y_j = \perp$, output $\perp$.
4. Otherwise, let $\mathbf{y} = (y_1, \ldots, y_n)$ and output $\mathsf{Decode}(\mathbf{y})$.

Theorem 63 (Integrity of Construction 5)**.** If PIR_0 is secure and provides ϵ-integrity and ECC is an error-correcting code that can correct up to t errors, then Construction 5 (instantiated with PIR_0 and ECC) provides ϵ^{t+1}-integrity.

Proof. Take any database $\mathbf{x} \in (\mathbb{F}^k)^N$, an index $i \in [N]$, and any efficient adversary $\mathcal{A}$. Write $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and let $\mathbf{y}_i \leftarrow \mathsf{Encode}(\mathbf{x}_i)$ for each $i \in [n]$. Let $\mathbf{z}_j \leftarrow (y_{1,j}, \ldots, y_{n,j})$ and let $d \leftarrow \mathsf{Digest}(1^\lambda, \mathbf{x})$. Then $d = (d_1, \ldots, d_n)$ where $d_j \leftarrow \mathsf{Digest}_0(1^\lambda, \mathbf{z}_j)$. Let $(\mathsf{st}, q) \leftarrow \mathsf{Query}(d, i)$ where $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_n)$, $q = (q_1, \ldots, q_n)$, and $(\mathsf{st}_j, q_j) \leftarrow \mathsf{Query}_0(d_j, i)$. Let $a^* = (a_1^*, \ldots, a_n^*)$ be the adversary's response in the integrity experiment. Let $y'_j \leftarrow \mathsf{Reconstruct}_0(\mathsf{st}_j, a_j^*)$. Consider now the output of $\mathbf{x}' \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$:

- Suppose there exists $j \in [t]$ such that $y'_j = \perp$. Then $\mathbf{x}' = \perp$.
- Suppose $y'_j = y_{i,j}$ for all but at most t indices $j \in [n]$. Since ECC can correct up to t errors, $\mathbf{x}' = \mathsf{Decode}((y'_1, \ldots, y'_j)) = \mathbf{x}_i$.
- Suppose there are at least $t+1$ indices $j \in [n]$ where $y'_j \notin \{y_{i,j}, \perp\}$. By integrity of PIR_0, for each $j \in [n]$,

  $$\Pr\left[y'_j \notin \{y_{i,j}, \perp\}\right] \leq \epsilon(\lambda) + \mathsf{negl}(\lambda).$$

  Moreover, this probability is taken only over the choice of the query randomness q_j. Since the queries $q_1, \ldots, q_n$ are sampled independently, the probability that there exists $t+1$ indices j where $y'_j \notin \{y_{i,j}, \perp\}$ is at most $\epsilon^{t+1} + \mathsf{negl}(\lambda)$.

By the above analysis, we conclude that

$$\Pr[\mathbf{x}' \notin \{\mathbf{x}_i, \perp\}] \leq \epsilon^{t+1} + \mathsf{negl}(\lambda),$$

and the claim holds. □

Theorem 64 (Privacy of Construction 5)**.** If PIR_0 provides privacy, then Construction 5 (instantiated with PIR_0) provides privacy.

Proof. Take any database $\mathbf{x} \in (\mathbb{F}^k)^N$, an index $i \in [N]$, and any efficient adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. Let $\mathcal{S}' = (\mathcal{S}'_0, \mathcal{S}'_1)$ be the simulator for PIR_0. We use $(\mathcal{S}'_0, \mathcal{S}'_1)$ to construct a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ for the transformed scheme:

Simulator $\mathcal{S}_0(1^\lambda, d, \mathbf{x})$

parse d as $(d_1, \ldots, d_n)$
parse $\mathbf{x}$ as $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$
$\mathbf{y}_i \leftarrow \mathsf{Encode}(\mathbf{x}_i)\ \forall i \in [N]$
for all $j \in [n]$:
  $\mathbf{z}_j \leftarrow (y_{1,j}, \ldots, y_{n,j})$
  $(\mathsf{st}_j, q_j) \leftarrow \mathcal{S}'_0(1^\lambda, d_j, \mathbf{z}_j)$
$\mathsf{st}_\mathcal{S} \leftarrow (\mathsf{st}_1, \ldots, \mathsf{st}_n)$
$q \leftarrow (q_1, \ldots, q_n)$
return $(\mathsf{st}_\mathcal{S}, q)$

Simulator $\mathcal{S}_1(\mathsf{st}_\mathcal{S}, a^*)$

parse $\mathsf{st}_\mathcal{S}$ as $(q_1, \ldots, q_n)$
parse a^* as $(a_1^*, \ldots, a_n^*)$
$b_j \leftarrow \mathcal{S}'_1(\mathsf{st}_j, a_j^*)\ \forall j \in [n]$
$b \leftarrow \mathbb{1}\{\forall j \in [n] : b_j = 1\}$
return b

We show that the real distribution $\text{REAL}_{\mathcal{A},\mathbf{x},i,\lambda}$ and ideal distribution $\text{IDEAL}_{\mathcal{A},\mathcal{S},\mathbf{x},\lambda}$ are computationally indistinguishable. We define a sequence of hybrid experiments:

- H_0: This is the real distribution $\text{REAL}_{\mathcal{A},\mathbf{x},i,\lambda}$:
  - The challenger starts by parsing $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and computes $\mathbf{y}_i \leftarrow \mathsf{Encode}(\mathbf{x}_i)$ for each $i \in [N]$. Then it forms $\mathbf{z}_j = (y_{1,j}, \ldots, y_{n,j})$ for each $j \in [n]$. It computes $d_j \leftarrow \mathsf{Digest}_0(1^\lambda, \mathbf{z}_j)$ and sets $d_j = (d_1, \ldots, d_n)$.
  - The challenger then samples $(\mathsf{st}_j, q_j) \leftarrow \mathsf{Query}_0(d_j, \mathbf{z}_j)$. It define $q = (q_1, \ldots, q_n)$ and gives $(d, \mathbf{x}, q)$ to $\mathcal{A}$.
  - The adversary responds with $a^* = (a_1^*, \ldots, a_n^*)$. For each $j \in [n]$, the challenger computes $y_j \leftarrow \mathsf{Reconstruct}_0(\mathsf{st}_j, a_j^*)$.
  - Then it computes $b \leftarrow \mathbb{1}\{\forall j \in [n] : y_j \neq \perp\}$ and gives b to $\mathcal{A}$.
  - The output of the experiment is $\mathcal{A}$'s output.
- H_1: Same as H_0 except after the challenger computes $y_1, \ldots, y_n$ from a^*, the challenger computes $b_j \leftarrow \mathbb{1}\{y_j \neq \perp\}$. Then, it sets $b \leftarrow \mathbb{1}\{\forall j \in [n] : b_j = 1\}$.
- H_2: Same as H_1 except the challenger computes $(\mathsf{st}_j, q_j) \leftarrow \mathcal{S}'_0(1^\lambda, d_j, \mathbf{z}_j)$ for each $j \in [N]$. After the adversary responds with $a^* = (a_1^*, \ldots, a_n^*)$, the challenger computes b_j as $b_j \leftarrow \mathcal{S}'_1(\mathsf{st}_j, a_j^*)$. This is the ideal distribution $\text{IDEAL}_{\mathcal{A},\mathcal{S},\mathbf{x},\lambda}$.

The difference between H_0 and H_1 is syntactic and their outputs are identically distributed. Hybrid H_1 and H_2 are computationally indistinguishable by security of PIR_0; formally, this follows by a sequence of n hybrid experiments where in experiment j, we switch to using the PIR_0 simulator $\mathcal{S}'$ to simulate the query q_j and the response bit b_j. □

B.2 Multi-server authenticated PIR for predicate queries

In this section we analyze our multi-server authenticated-PIR scheme for predicate queries.

B.2.1 Security proofs

We prove security only for the case of $k = 2$ servers. All the arguments generalize naturally to the k-server setting with $k > 2$. Correctness of the multi-server authenticated PIR scheme for predicate queries introduced in Construction 2 can be verified by inspection. To prove integrity and security, we find it useful to first prove Lemma 65, which states that if an adversary deviates from the prescribed protocol, the $\mathsf{Reconstruct}$ algorithm rejects with high probability.

Lemma 65. Let the authenticated PIR scheme introduced in Construction 2, where $k = 2$ for this lemma. Then, for every database size $N \in \mathbb{N}$, for every non-zero offset $\boldsymbol{\Delta} = (\Delta_m, \Delta_\tau) \in \mathbb{F}^2$, every database $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$, every vector of weights $\mathbf{w} \in \mathbb{F}^N$, and function $f \in \mathcal{F}$, we have

$$\Pr\left[y \neq \perp : \begin{array}{r} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(1^\lambda, f) \\ \mathbf{a}_1 \leftarrow \mathsf{Answer}(\mathbf{X}, \mathbf{w}, q_1) \\ \mathbf{a}_2 \leftarrow \mathsf{Answer}(\mathbf{X}, \mathbf{w}, q_2) \\ y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a_1 + \Delta, a_2) \end{array}\right] \leq \frac{1}{|\mathbb{F}| - 1},$$

where the probability is computed over all the random coins used by the algorithms of the experiment. The statement holds also when the $\mathsf{Reconstruct}$ algorithm instead takes as input $(\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2 + \Delta)$.

Proof. Let $\alpha \leftarrow\!\!\$\, \mathbb{F} \setminus \{0\}$. By construction, we can rewrite the probability stated in the lemma as

$$\begin{aligned} \nu &= \Pr\left[\alpha \cdot \left(\sum_{i \in [N]} w_i \cdot f(i, \mathbf{x}_i) + \Delta_m\right) = \alpha \cdot \sum_{i \in [N]} w_i \cdot f(i, \mathbf{x}_i) + \Delta_\tau\right] \\ &= \Pr\left[-\Delta_\tau + \alpha \cdot \Delta_m = 0\right] \end{aligned}$$

The last quantity is the evaluation of a non-zero degree-1 polynomial with coefficients Δ_τ and Δ_m at a random point $\alpha \leftarrow\!\!\$\, \mathbb{F} \setminus \{0\}$. Since a non-zero linear polynomial has at most one root over $\mathbb{F} \setminus \{0\}$, we conclude that $\nu \leq \frac{1}{|\mathbb{F}|-1}$. By interchanging the roles of $\mathbf{a}_1$ and $\mathbf{a}_2$, the statement holds also when the $\mathsf{Reconstruct}$ algorithm instead takes as input $(\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2 + \Delta)$. □

We now use Lemma 65 to show that the scheme presented in Construction 2 ensures

integrity and security, and hence it is secure.

Theorem 66 (Integrity of Construction 2)**.** The authenticated PIR scheme of Construction 2 provides integrity.

Proof. This theorem follows directly from Lemma 65. □

Theorem 67 (Privacy of Construction 2)**.** The authenticated PIR scheme of Construction 2 provides privacy.

Proof. The proofs proceeds exactly as the proof for Theorem 45, with the difference that we use the simulator induced by the secure function-secret-sharing scheme instead of the simulator induced by the classic PIR scheme, and we appeal to Lemma 65 instead of Lemma 43 to conclude the proof. □

B.2.2 Handling functions with larger output

In this section we discuss how to handle functions with larger output in authenticated PIR for predicate queries.

Scheme

The scheme is described in Construction 6.

Security analysis

Lemma 68. Let the authenticated PIR scheme introduced in Construction 6, where $k = 2$ for this lemma. Then, for every database size $N \in \mathbb{N}$, for every non-zero vector $(\Delta_0, \ldots, \Delta_b) \in \mathbb{F}^{b+1}$, every database $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$, every vector of weights $\mathbf{w} \in \mathbb{F}^N$, and every function $f \in \mathcal{F}$, the following holds:

$$\Pr\left[y \neq \perp : \begin{array}{r} (\mathsf{st}, q_1, q_2) \leftarrow \mathsf{Query}(\lambda, f) \\ \mathbf{a}_1 \leftarrow \mathsf{Answer}(\mathbf{X}, \mathbf{w}, q_1) \\ \mathbf{a}_2 \leftarrow \mathsf{Answer}(\mathbf{X}, \mathbf{w}, q_2) \\ y \leftarrow \mathsf{Reconstruct}(\mathsf{st}, \mathbf{a}_1 + \Delta, \mathbf{a}_2) \end{array}\right] \leq \frac{b}{|\mathbb{F}| - 1},$$

where the probability is computed over all the random coins used by the algorithms of the scheme. Without loss of generality, the statement holds also when the roles of honest and malicious server are inverted.

Construction 6 (k-server authenticated PIR for predicate queries for functions whose output is larger than a single field element tolerating $k-1$ malicious servers)**.** The construction is parametrized by a number of servers $k \in \mathbb{N}$, a number of database rows $N \in \mathbb{N}$, a row length $\ell \in \mathbb{N}$, a finite field $\mathbb{F}$, a security parameter λ, a output length $b \in \mathbb{N}$, a function class $\mathcal{F} \subseteq \mathsf{Funs}[[N] \times \{0,1\}^\ell, \mathbb{F}^b]$ that is closed under scalar multiplication, and a function-secret-sharing scheme $(\mathsf{FSS.Gen}, \mathsf{FSS.Eval})$ for the function class $\mathcal{F}$, parametrized bx the security parameter λ. We represent the database as N binary strings of length ℓ each: $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell$.

$\mathsf{Query}\left(1^\lambda, f\right) \to (\mathsf{st}, \mathbf{q}_1, \ldots, \mathbf{q}_k)$

1. Sample a random field element $\alpha \xleftarrow{R} \mathbb{F} \setminus \{0\}$.
2. Set the state $\mathsf{st} \leftarrow \alpha$.
3. For $j \in [b]$, let $g_j \leftarrow \alpha^j \cdot f$. These functions g_j must exist since the function class $\mathcal{F}$ is closed under scalar multiplication, as in Definition 49.
4. Compute $q_1, \ldots, q_k \leftarrow \mathsf{FSS.Gen}(1^\lambda, f)$ together with $q_1^{(i)}, \ldots, q_k^{(i)} \leftarrow \mathsf{FSS.Gen}(1^\lambda, g_i)$, for $i \in [b]$.
5. Output $\left(\mathsf{st}, (q_1, q_1^{(1)}, \ldots, q_1^{(b)}), \ldots, (q_k, q_k^{(1)}, \ldots, q_k^{(b)})\right)$.

$\mathsf{Answer}\left(\mathbf{x}_1, \ldots, \mathbf{x}_N \in \{0,1\}^\ell, \mathbf{w} \in \mathbb{F}^N, \mathbf{q}\right) \to \mathbf{a} \in \mathbb{F}^{b+1}$

1. Parse $\mathbf{q}$ as $(q_f, q_g^{(1)}, \ldots, q_g^{(b)})$.
2. Compute answer as $a_f \leftarrow \sum_{i \in [N]} w_i \cdot \mathsf{FSS.Eval}(q_f, x_i)$ and $a_g \leftarrow \sum_{j \in [b]} \left(\sum_{i \in [N]} w_i \cdot \mathsf{FSS.Eval}\left(q_g^{(j)}, x_i\right)\right)$.
3. Return $\mathbf{a} \leftarrow (a_f, a_g)$.

$\mathsf{Reconstruct}\left(\mathsf{st}, a_1, \ldots, a_k \in \mathbb{F}^{b+1}\right) \to \mathbb{F}^b \cup \{\perp\}$

1. Parse the state st as $\alpha \in \mathbb{F}$.
2. Compute $\mathbf{a} \leftarrow a_1 + \cdots + a_k \in \mathbb{F}^{b+1}$.
3. Parse a as $(m_1, \ldots, m_b, \tau) \in \mathbb{F}^{b+1}$.
4. Compute $\tau' \leftarrow m_1\alpha + m_2\alpha^2 + \cdots + m_b\alpha^b \in \mathbb{F}$.
5. If $\tau = \tau'$, output $(m_1, \ldots, m_b) \in \mathbb{F}^b$. Otherwise, output $\perp$.

Proof. Let $\alpha \leftarrow\!\!\$\, \mathbb{F} \setminus \{0\}$. Let

$$y = (m_1, \ldots, m_b) \leftarrow \sum_{i \in [N]} w_i \cdot f(i, \mathbf{x}_i) \in \mathbb{F}^b.$$

Then the probability stated in the lemma is

$$\begin{aligned} \nu &= \Pr\left[\sum_{j \in [b]} (m_j + \Delta_j)\alpha^j = \Delta_0 + \sum_{j \in [b]} m_j \alpha^j\right] \\ &= \Pr\left[-\Delta_0 + \sum_{j \in [b]} \Delta_j \alpha^j = 0\right]. \end{aligned}$$

This last quantity is the evaluation of a non-zero polynomial (whose coefficients are the Δ values) at a random point $\alpha \xleftarrow{R} \mathbb{F} \setminus \{0\}$. Since such a non-zero polynomial of degree at most b can have at most b roots over $\mathbb{F}$, we have that $\nu \leq \frac{b}{|\mathbb{F}|-1}$. By interchanging the roles of a_0 and a_1, the statement holds also when the Reconstruct algorithm instead takes as input $(\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2 + \Delta)$. □

Theorem 69 (Integrity of Construction 6)**.** The authenticated PIR scheme of Construction 6 provides integrity.

Proof. The theorem follows directly from Lemma 68. □

Theorem 70 (Security of Construction 6)**.** The authenticated PIR scheme of Construction 6 provides privacy.

Proof. The strategy is as in the proof of Theorem 67, except that we appeal to Lemma 68 to complete the argument. □

B.3 Security proofs for single-server authenticated PIR from LWE

In this section we provide the security proofs for Construction 3.

Theorem 71 (Correctness of Construction 3)**.** If $B \geq \sqrt{\lambda N} s$, then Construction 3 is correct.

Proof. Take any database $x \in \{0,1\}^N$ and index $i \in [N]$. Let $\mathbf{d} = \mathbf{A}\mathbf{x}$ be the digest,

$\mathbf{q}^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T} + t \cdot \boldsymbol{\eta}_i^\mathsf{T}$ be the query, and $a \leftarrow \mathbf{q}^\mathsf{T}\mathbf{x}$ be the response. Then, we have

$$\begin{aligned} a - \mathbf{s}^\mathsf{T}\mathbf{d} - x_i t &= \mathbf{q}^\mathsf{T}\mathbf{x} - \mathbf{s}^\mathsf{T}\mathbf{d} - x_i t \\ &= \mathbf{s}^\mathsf{T}\mathbf{A}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} + t \cdot \boldsymbol{\eta}_i^\mathsf{T}\mathbf{x} - \mathbf{s}^\mathsf{T}\mathbf{A}\mathbf{x} - x_i t \\ &= \mathbf{e}^\mathsf{T}\mathbf{x}. \end{aligned} \tag{B.1}$$

Since the components of $\mathbf{e}$ are independent discrete Gaussian random variables with parameter s, $\mathbf{e}^\mathsf{T}\mathbf{x}$ is subgaussian with parameter $\|\mathbf{x}\| \cdot s \leq \sqrt{N}s$ since $\mathbf{x} \in \{0,1\}^N$. By Eq. (3.1),

$$\begin{aligned} \Pr[|\mathbf{e}^\mathsf{T}\mathbf{x}| < B : \mathbf{e} \leftarrow D_{\mathbb{Z},s}^N] &\geq \Pr[|\mathbf{e}^\mathsf{T}\mathbf{x}| \leq \sqrt{\lambda N}s : \mathbf{e} \leftarrow D_{\mathbb{Z},s}^N] \\ &= 1 - \mathsf{negl}(\lambda). \end{aligned} \tag{B.2}$$

To complete the proof, we show that $|a - \mathbf{s}^\mathsf{T}\mathbf{d} - (1 - x_i)t| \geq B$. By Eq. (B.1),

$$|a - \mathbf{s}^\mathsf{T}\mathbf{d} - (1 - x_i)t| = |\mathbf{e}^\mathsf{T}\mathbf{x} + (1 - 2x_i)t|.$$

By Eq. (3.1), $|\mathbf{e}^\mathsf{T}\mathbf{x}| < B$ with overwhelming probability. Since $1 - 2x_i \in \{-1, 1\}$ and $t \in [2B, q - 2B]$, with overwhelming probability over the choice of $\mathbf{e}$, we have $\mathbf{e}^\mathsf{T}\mathbf{x} + (1 - 2x_i)t \in [B, q - B]$, or equivalently, $|\mathbf{e}^\mathsf{T}\mathbf{x} + (1 - 2x_i)t| \geq B$. □

Before proving security and privacy, we first prove the following lemma, similar to the approach used for multi-server schemes.

Lemma 72. Let λ be a security parameter, $\mathbf{x} \in \{0,1\}^N$ be a database, $i \in [N]$ be an index, and $\mathcal{A}$ be an adversary. Consider Construction 3 and define distributions $D^{(0)}_{\mathcal{A},x,i,\lambda}$, $D^{(1)}_{\mathcal{A},x,i,\lambda}$:

Distribution $D^{(0)}_{\mathcal{A},x,i,\lambda}$

$\mathbf{d} \leftarrow \mathsf{Digest}(1^\lambda, \mathbf{x})$
$(\mathsf{st}, \mathbf{q}) \leftarrow \mathsf{Query}(\mathbf{d}, i)$
$(\mathsf{st}_\mathcal{A}, a^*) \leftarrow \mathcal{A}(\mathbf{d}, \mathbf{x}, \mathbf{q})$
$x_i' \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$
return x_i'

Distribution $D^{(1)}_{\mathcal{A},x,i,\lambda}$

$\mathbf{d} \leftarrow \mathsf{Digest}(1^\lambda, \mathbf{x})$
$\mathbf{q} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^N, \mathbf{e} \leftarrow D_{\mathbb{Z},s}^N$
$(\mathsf{st}_\mathcal{A}, a^*) \leftarrow \mathcal{A}(\mathbf{d}, \mathbf{x}, \mathbf{q})$
$t \xleftarrow{\mathrm{R}} [2B, q - 2B]$
$\mathbf{u}^\mathsf{T} \leftarrow \mathbf{q}^\mathsf{T} - t \cdot \boldsymbol{\eta}_i^\mathsf{T}$
$\hat{a}^* \leftarrow a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x}$
if $|\hat{a}^*| < B$ **then**
  $x_i' \leftarrow 0$
elseif $|\hat{a}^* - t| < B$ **then**
  $x_i' \leftarrow 1$
else $x_i' \leftarrow \perp$
return x_i'

Suppose the $\mathsf{extLWE}_{n,N,q,s}$ assumption holds and H is modeled as a random oracle. Then, for every database length $N = N(\lambda)$, database $x \in \{0,1\}^N$, index $i \in [N]$, and every adversary $\mathcal{A}$ running in time $t = t(\lambda)$, there exists an adversary $\mathcal{B}$ running in time $\mathsf{poly}(t)$ such that

$$|\Pr[D^{(0)}_{\mathcal{A},x,i,\lambda} = 1] - \Pr[D^{(1)}_{\mathcal{A},x,i,\lambda} = 1]| \leq \mathsf{Adv}^{(n,N,q,s)}_{\mathsf{extLWE}}[\mathcal{B}].$$

Proof. Fix a database $\mathbf{x} \in \{0,1\}^N$, an index $i \in [N]$, and any efficient adversary $\mathcal{A}$. In the following analysis, we write $\mathbf{a}_i \in \mathbb{Z}_q^n$ to denote $H(i)$ and we model H as a random oracle (which the reduction algorithm is allowed to program [BR93]). We now define a sequence of hybrid experiments:

- H_0: This is the distribution $D^{(0)}_{\mathcal{A},\mathbf{x},i,\lambda}$. In this distribution, the output x'_i is computed via $x'_i \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$.
- H_1: Same as H_0 except the challenger changes how x'_i is computed. Instead of computing $x'_i \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$, the challenger sets x'_i as follows:
  - If $|a^* - (\mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T})\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - kt| < B$ for $k \in \{0,1\}$, then $x'_i \leftarrow k$.
  - Otherwise, the challenger sets $x'_i \leftarrow \perp$.
- H_2: Same as H_1 except the challenger replaces $\mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T}$ with a uniform random vector $\mathbf{u}^\mathsf{T} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^N$. Specifically, the challenger computes $\mathbf{q}^\mathsf{T} \leftarrow \mathbf{u}^\mathsf{T} + t \cdot \boldsymbol{\eta}_i^\mathsf{T}$ and x'_i as follows:
  - If $|a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - kt| < B$ for $k \in \{0,1\}$, then $x'_i \leftarrow k$.
  - Otherwise, the challenger sets $x'_i \leftarrow \perp$.
- H_3: Same as H_2 except the challenger samples $\mathbf{q} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^N$. Then, *after* the adversary outputs the response a^*, it samples $t \xleftarrow{\mathrm{R}} [2B, q-2B]$ and sets $\mathbf{u}^\mathsf{T} \leftarrow \mathbf{q}^\mathsf{T} - t \cdot \boldsymbol{\eta}_i^\mathsf{T}$. The response a'_i is computed exactly as in H_2. This is the distribution $D^{(1)}_{\mathcal{A},x,i,\lambda}$.

To complete the proof, we now show that each adjacent pair of distributions is indistinguishable.

- Hybrids H_0 and H_1 are identical distributions. In both experiments, $\mathbf{d} = \mathbf{A}\mathbf{x}$, $\mathbf{q}^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T} + t \cdot \boldsymbol{\eta}_i^\mathsf{T}$ and $\mathsf{st} = (\mathbf{d}, \mathbf{s}, t)$. Let a^* be the adversary's response in H_0 and consider the value of $x'_i \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$. Let $z = a^* - \mathbf{s}^\mathsf{T}\mathbf{d}$. Then,

$$\begin{aligned} z = a^* - \mathbf{s}^\mathsf{T}\mathbf{d} &= a^* - \mathbf{s}^\mathsf{T}\mathbf{A}\mathbf{x} \\ &= a^* - (\mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T})\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} \end{aligned}$$

In H_0, the challenger outputs $k \in \{0,1\}$ if $|a^* - \mathbf{s}^\mathsf{T}\mathbf{d} - kt| = |z - kt| < B$ and $\perp$ otherwise. By the above calculation, this precisely coincides with the procedure in H_1.

- Hybrids H_1 and H_2 are computationally indistinguishable under the $\mathsf{extLWE}_{n,N,q,s}$ assumption and modeling H as a random oracle. To see this, suppose there exists an efficient adversary $\mathcal{A}$ that is able to distinguish hybrids H_1 and H_2 with non-negligible advantage. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the extended LWE assumption:

  1. At the beginning of the game, algorithm $\mathcal{B}$ receives an extended LWE challenge $(\mathbf{A}, \mathbf{z}^\mathsf{T}, y)$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times N}$, $\mathbf{z} \in \mathbb{Z}_q^N$, and $y \in \mathbb{Z}_q$.
  2. Let $\mathbf{a}_1, \ldots, \mathbf{a}_N \in \mathbb{Z}_q^n$ be the columns of $\mathbf{A}$. Algorithm $\mathcal{B}$ programs the random oracle $H(i) \mapsto \mathbf{a}_i$ for each $i \in [N]$. If $\mathcal{A}$ ever queries H on an input $k \notin N$, algorithm $\mathcal{B}$ samples a random $\mathbf{r}_k \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and defines the mapping $H(k) \mapsto \mathbf{r}_k$.
  3. Algorithm $\mathcal{B}$ now constructs the digest $\mathbf{d} \leftarrow \mathbf{A}\mathbf{x}$ as in H_1 and H_2. To construct the query, algorithm $\mathcal{B}$ samples $t \xleftarrow{\mathrm{R}} [2B, q - 2B]$ and sets $\mathbf{q}^\mathsf{T} \leftarrow \mathbf{z}^\mathsf{T} + t \cdot \boldsymbol{\eta}_i^\mathsf{T}$. It gives the digest $\mathbf{d}$, the database $\mathbf{x}$, and the query $\mathbf{q}$ to $\mathcal{A}$.
  4. Algorithm $\mathcal{A}$ outputs a response a^*. Algorithm $\mathcal{B}$ computes x_i' as follows:
     - If $|a^* - \mathbf{z}^\mathsf{T}\mathbf{x} + y - kt| < B$ for $k \in \{0,1\}$, then $x_i' \leftarrow k$.
     - Otherwise, $x_i' \leftarrow \perp$.
  5. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with x_i' and outputs whatever $\mathcal{A}$ outputs.

  Since $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times N}$, the outputs of the random oracle are correctly simulated. Corresponding, algorithm $\mathcal{B}$ perfectly simulates the distribution of the digest $\mathbf{d}$ for $\mathcal{A}$. We now consider the two possible challenge distributions:

  - Suppose $\mathbf{z}^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T}$ and $y = \mathbf{e}^\mathsf{T}\mathbf{x}$. Then the query $\mathbf{q}$ and the response x_i' are distributed exactly as in H_1.
  - Suppose $\mathbf{z}^\mathsf{T} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^N$ and $y = \mathbf{e}^\mathsf{T}\mathbf{x}$. Then, the query $\mathbf{q}$ and the response x_i' are distributed exactly as in H_2.

  We conclude that algorithm $\mathcal{B}$ breaks the extended LWE assumption with the same distinguishing advantage as $\mathcal{A}$ and the claim follows. More precisely, we can write $\mathrm{H}_i(\mathcal{A})$ to denote the output of a distinguisher $\mathcal{A}$ on input a sample from H_i. Then our reduction shows that for all adversaries $\mathcal{A}$ running in time t, there exists an adversary $\mathcal{B}$ running in time $\mathsf{poly}(t)$ such that

$$\mathsf{Adv}^{n,N,q,s}_{\mathsf{extLWE}}[\mathcal{B}] \geq |\Pr[\mathrm{H}_1[\mathcal{A}] = 1] - \Pr[\mathrm{H}_2[\mathcal{A}] = 1]|.$$

- Hybrids H_2 and H_3 are identically distributed. In H_2, $\mathbf{q} = \mathbf{u} + t \cdot \boldsymbol{\eta}_i$ where $\mathbf{u} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^N$ and $\mathbf{u}$ is sampled independently of all other quantities. Thus, the distribution of $\mathbf{q}$ in H_2 is uniform over $\mathbb{Z}_q^N$, which matches the distribution in H_3. In both experiments, $\mathbf{u} = \mathbf{q} - t \cdot \boldsymbol{\eta}_i$, where $t \xleftarrow{\mathrm{R}} [2B, q - 2B]$. □

Theorem 73 (Integrity of Construction 3)**.** Suppose the $\mathsf{extLWE}_{n,N,q,s}$ assumption holds and H is modeled as a random oracle. Then, Construction 3 (instantiated with parameters n, N, q, s, B and hash function H) has integrity error at most $\epsilon = (2B-1)/(q-4B+1)$.

Proof. Fix a database $\mathbf{x} \in \{0,1\}^N$, an index $i \in [N]$, and any efficient adversary $\mathcal{A}$. We now define a sequence of hybrid experiments:

- H_0: This is the real integrity game.
- H_1: Same as H_0 except the challenger samples $\mathbf{q} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^N$ and $\mathbf{e} \leftarrow D_{\mathbb{Z},s}^N$. Then, after the adversary outputs the response a^*, the challenger samples $t \xleftarrow{\mathrm{R}} [2B, q-2B]$ and sets $\mathbf{u}^\mathsf{T} \leftarrow \mathbf{q}^\mathsf{T} - t \cdot \boldsymbol{\eta}_i^\mathsf{T}$. If $|a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - kt| < B$ for some $k \in \{0,1\}$, then the challenger sets $x_i' \leftarrow k$. Otherwise, the challenger sets $x_i' \leftarrow \perp$.
- H_2: Same as H_1 except the challenger changes how it computes x_i':
  - If $|a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - x_i t| < B$, then $x_i' \leftarrow x_i$.
  - Otherwise, the challenger sets $x_i' \leftarrow \perp$.

  Specifically, in H_2, it is guaranteed that $x_i' \in \{x_i, \perp\}$.

We now show that the outputs of each adjacent pair of hybrid distributions are computationally indistinguishable:

- Hybrids H_0 and H_1 are computationally indistinguishable by Lemma 72.
- The statistical distance between H_1 and H_2 is at most $(2B+1)/(q-4B+1)$. By construction, the two experiments are identical unless

$$|a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - (1-x_i)t| < B. \tag{B.3}$$

  Now, $\mathbf{u} = \mathbf{q} - t \cdot \boldsymbol{\eta}_i$, so

$$a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - (1-x_i)t = a^* - \mathbf{q}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - (1-2x_i)t.$$

  Since $1-2x_i \in \{-1,1\}$, there are at most $2B-1$ values of $t \in \mathbb{Z}_q$ for which Eq. (B.3) holds. Since t is sampled uniformly at random from a set of size $q-4B+1$ and independently of a^*, $\mathbf{u}$, $\mathbf{x}$, and $\mathbf{e}$, the probability that t lands in the interval of size $2B-1$ is at most $(2B-1)/(q-4B+1)$. Correspondingly, the statistical distance between H_1 and H_2 is $(2B-1)/(q-4B+1)$.

By construction, the output x_i' in H_2 is guaranteed to be either x_i or $\perp$. By a hybrid argument, in the real integrity game H_0, it must be the case that

$$\Pr[x_i' \in \{x_i, \perp\}] \leq \frac{2B-1}{q-4B+1} + \mathsf{negl}(\lambda),$$

which proves the claim. □

Theorem 74 (Privacy of Construction 3)**.** Suppose the $\mathsf{extLWE}_{n,N,q,s}$ assumption holds and H is modeled as a random oracle. Then, Construction 3 (instantiated with parameters n, N, q, s, B and hash function H) provides privacy. More precisely, for every adversary running in time $t = t(\lambda)$, there exists an adversary $\mathcal{B}$ running in time $\mathsf{poly}(t)$ such that

$$|\Pr[\text{REAL}_{\mathcal{A},x,i,\lambda} = 1] - \Pr[\text{IDEAL}_{\mathcal{A},\mathcal{S},x,\lambda} = 1]| \leq \mathsf{Adv}^{n,N,q,s}_{\mathsf{extLWE}}[\mathcal{B}],$$

where $\text{REAL}_{\mathcal{A},x,i,\lambda}$ and $\text{IDEAL}_{\mathcal{A},\mathcal{S},x,\lambda}$ are the distributions defined in Definition 32.

Proof. Fix a database $\mathbf{x} \in \{0,1\}^N$, an index $i \in [N]$, and any efficient adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. We construct an efficient simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ as follows:

Simulator $\mathcal{S}_0(1^\lambda, \mathbf{d}, \mathbf{x})$	**Simulator** $\mathcal{S}_1(\mathsf{st}_\mathcal{S}, a^*)$
$\mathbf{q} \xleftarrow{R} \mathbb{Z}_q^N, \mathbf{e} \leftarrow D_{\mathbb{Z},s}^N$	**if** $\lvert a^* - \mathsf{st}_\mathcal{S}\rvert < B$ **then** $b \leftarrow 1$
$\mathsf{st}_\mathcal{S} \leftarrow \mathbf{q}^\mathsf{T}\mathbf{x} - \mathbf{e}^\mathsf{T}\mathbf{x}$	**else**
return $(\mathsf{st}_\mathcal{S}, \mathbf{q})$	$t \xleftarrow{R} [2B, q - 2B]$
	$b \leftarrow \mathbb{1}\{\lvert a^* - \mathsf{st}_\mathcal{S} - t\rvert < B\}$
	return b

We show that the real distribution $\text{REAL}_{\mathcal{A},\mathbf{x},i,\lambda}$ and ideal distribution $\text{IDEAL}_{\mathcal{A},\mathcal{S},\mathbf{x},\lambda}$ are computationally indistinguishable. We define a sequence of hybrid experiments:

- H_0: This is the real distribution $\text{REAL}_{\mathcal{A},\mathbf{x},i,\lambda}$. In this distribution, the response x'_i is computed via $x'_i \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$.
- H_1: Same as H_0 except the challenger samples $\mathbf{q} \xleftarrow{R} \mathbb{Z}_q^N$ and $\mathbf{e} \leftarrow D_{\mathbb{Z},s}^N$. Then, after the adversary outputs the response a^*, the challenger samples $t \xleftarrow{R} [2B, q - 2B]$ and sets $\mathbf{u}^\mathsf{T} \leftarrow \mathbf{q}^\mathsf{T} - t \cdot \boldsymbol{\eta}_i^\mathsf{T}$. If $|a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - kt| < B$ for some $k \in \{0,1\}$, then the challenger sets $x'_i \leftarrow k$. Otherwise, the challenger sets $x'_i \leftarrow \perp$.
- H_2: Same as H_1, except instead of computing x'_i, the challenger sets $b = 1$ if $|a^* - \mathbf{q}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x}| < B$. Otherwise, it samples $t \xleftarrow{R} [2B, q - 2B]$ and sets $b \leftarrow \mathbb{1}\{|a^* - \mathbf{q}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - t| < B\}$ This is the ideal distribution $\text{IDEAL}_{\mathcal{A},\mathcal{S},\mathbf{x},\lambda}$.

To complete the proof, we now show that each adjacent pair of distributions is indistinguishable. First, hybrids H_0 and H_1 are computationally indistinguishable by Lemma 72. To complete the proof, we show that H_1 and H_2 are identically distributed:

- Hybrids H_1 and H_2 are identically distributed. Let $a^* \in \mathbb{Z}_q$ be the adversary's response in the two experiments. Define the quantity $z = a^* - \mathbf{q}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x}$. We consider two possibilities:

  – Suppose $|z| < B$. In H_2, the challenger always sets $b = 1$. We claim this is also the case in H_1. By construction, we can first write

  $$\mathbf{u}^\mathsf{T}\mathbf{x} = \mathbf{q}^\mathsf{T}\mathbf{x} - t \cdot \boldsymbol{\eta}_i^\mathsf{T}\mathbf{x} = \mathbf{q}^\mathsf{T}\mathbf{x} - x_i t. \tag{B.4}$$

  This means

  $$|z| = |a^* - \mathbf{q}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x}| = |a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - x_i t|. \tag{B.5}$$

  Since $x_i \in \{0, 1\}$, we have $x_i' = x_i$ and $b = 1$ in H_1.

  – Suppose $|z| \geq B$. In this case, the challenger in H_2 samples $t \xleftarrow{R} [2B, q - 2B]$ and sets $b = 1$ if $|z - t| < B$ and $b = 0$ otherwise. Consider the challenger's behavior in H_1. By Eq. (B.5), we have that $b = 1$ only if

  $$|a^* - \mathbf{u}^\mathsf{T}\mathbf{x} + \mathbf{e}^\mathsf{T}\mathbf{x} - (1 - x_i)t| < B.$$

  By Eq. (B.4), this is equivalent to $|z - (1 - 2x_i)t| < B$. Like in H_2, the challenger in H_1 samples $t \xleftarrow{R} [2B, q - 2B]$ *after* the adversary outputs a^*. We consider two possibilities:

  * If $x_i = 0$, then $1 - 2x_i = 1$, and the challenger in H_1 sets $b \leftarrow \mathbb{1}\{|z - t| < B\}$. This is identical to the behavior in H_2.
  * If $x_i = 1$, then $1 - 2x_i = -1$, and the challenger in H_1 sets $b \leftarrow \mathbb{1}\{|z + t| < B\}$. Since $t \xleftarrow{R} [2B, q - 2B]$ the distributions of $t \bmod q$ and $-t \bmod q$ are identical (the interval is symmetric about 0 over $\mathbb{Z}_q$). Since t and z are independent, the distribution of $\mathbb{1}\{|z + t| < B\}$ is identically distributed as that of $\mathbb{1}\{|z - t| < B\}$. Once again, the distribution of b in H_1 is distributed identically to that in H_2.

  We conclude that the distribution of b is identical in H_1 and H_2 in this case. □

B.4 Single-server authenticated PIR from DDH

In this section we provide supplementary material on the single-server authenticated PIR scheme from the decisional Diffie-Hellman assumption (Construction 4).

B.4.1 Security proofs

Correctness of Construction 4 can be verified by inspection.

To analyze the security and integrity of Construction 4, we start by proving the following lemma, which will feature in both the security and the integrity analysis.

Lemma 75. Let λ be a security parameter, $x \in \{0,1\}^N$ be a database, $i \in [N]$ be an index, and $\mathcal{A}$ be an adversary. Consider Construction 4 and define distributions $D^{(0)}_{\mathcal{A},x,i,\lambda}$ and $D^{(1)}_{\mathcal{A},x,i,\lambda}$:

Distribution $D^{(0)}_{\mathcal{A},x,i,\lambda}$	**Distribution** $D^{(1)}_{\mathcal{A},x,i,\lambda}$
$d \leftarrow \mathsf{Digest}(1^\lambda, x)$	$d \leftarrow \mathsf{Digest}(1^\lambda, x)$
$(\mathsf{st}, q) \leftarrow \mathsf{Query}(d, i)$	$q \xleftarrow{R} \mathbb{G}^N$
$(\mathsf{st}_{\mathcal{A}}, a^*) \leftarrow \mathcal{A}(d, x, q)$	$(\mathsf{st}_{\mathcal{A}}, a^*) \leftarrow \mathcal{A}(d, x, q)$
$x'_i \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$	**if** $a^* = \prod_{j \in [N]} q_j^{x_j}$ **then** $x'_i \leftarrow x_i$
return x'_i	**else** $x'_i \leftarrow \perp$
	return x'_i

Suppose the DDH assumption holds in $\mathbb{G}$ and H is modeled as a random oracle. Then, for every database length $N = N(\lambda)$, database $x \in \{0,1\}^N$, index $i \in [N]$, and efficient adversary $\mathcal{A}$,

$$\left|\Pr[D^{(0)}_{\mathcal{A},x,i,\lambda} = 1] - \Pr[D^{(1)}_{\mathcal{A},x,i,\lambda} = 1]\right| \leq \mathsf{negl}(\lambda).$$

Proof. Take any database length $N = N(\lambda)$, database $x \in \{0,1\}^N$ and an index $i \in [N]$. We show that the distributions $D^{(0)}_{\mathcal{A},x,i,\lambda}$ and $D^{(1)}_{\mathcal{A},x,i,\lambda}$ are computationally indistinguishable. In the following analysis, we write $h_i \in \mathbb{G}$ to denotes $H(i)$, and we model H as a random oracle (which the reduction algorithm is allowed to program) [BR93]. We now define a sequence of hybrid experiments:

- H_0: This is the distribution $D^{(0)}_{\mathcal{A},x,i,\lambda}$. In this distribution, the response $x'_i \in \{0,1,\perp\}$ is computed via $x'_i \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$.
- H_1: Same as H_0, except the challenger changes how x'_i is computed. Instead of computing $x'_i \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$, the challenger sets x'_i as follows:
  - If $a^* = h_i^{yt}(h_i^r)^{x_i} \prod_{j \neq i}(h_j^r)^{x_j}$ for $y \in \{0,1\}$, then $x'_i \leftarrow y$.
  - Otherwise, the challenger sets $x'_i \leftarrow \perp$.
- H_2: Same as H_1, except the challenger replaces the tuple of group elements $(g, h_1, h_1^r, \ldots, h_N, h_N^r)$ with $(g, h_1, z_1, \ldots, h_N, z_N)$ where $z_1, \ldots, z_N \xleftarrow{R} \mathbb{G}$ and $r \xleftarrow{R} \mathbb{Z}_p$. Specifically, the challenger constructs the query $q = (q_1, \ldots, q_N)$ by setting $q_j \leftarrow z_j$ for $j \neq i$ and $q_i \leftarrow z_i h_i^t$. When computing x'_i, the challenger proceeds as follows:

  - If $a^* = h_i^{yt} z_i^{x_i} \prod_{j \neq i} z_j^{x_j}$ for $y \in \{0,1\}$, then $x_i' \leftarrow y$.
  - Otherwise, the challenger sets $x_i' \leftarrow \perp$.

- H_3: Same as H_2 except the challenger samples $q \xleftarrow{\mathrm{R}} \mathbb{G}^N$. Then, *after* the adversary outputs the response a^*, it sets $z_j = q_j$ for all $j \neq i$ and $z_i \leftarrow q_i / h_i^t$ where $t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. The response a_i' is computed exactly as in H_2.

- H_4: Same as H_2 except the challenger again changes how it computes x_i':

  - If $a^* = h_i^{x_i t} z_i^{x_i} \prod_{j \neq i} z_j^{x_j}$, then $x_i' \leftarrow x_i$.
  - Otherwise, the challenger sets $x_i' \leftarrow \perp$.

- H_5: Same as H_4, except the challenger sets $x_i' \leftarrow x_i$ if $a^* = \prod_{j \in [N]} q_j^{x_j}$ and $x_i' \leftarrow \perp$ otherwise. This is the distribution $D^{(1)}_{\mathcal{A},x,i,\lambda}$.

To complete the proof, we now show that each adjacent pair of distributions are indistinguishable:

- Hybrids H_0 and H_1 are identical distributions. In both experiments, $d = \prod_{j \in [N]} h_j^{x_j}$, $q = (q_1, \ldots, q_N)$, and $\mathsf{st} = (i, d, r, t)$, where $q_j = h_j^r$ for $j \neq i$ and $q_i = h_i^{r+t}$ for some $r, t \in \mathbb{Z}_p$. Let a^* be the adversary's response in H_0, and consider the value of $x_i' \leftarrow \mathsf{Reconstruct}(\mathsf{st}, a^*)$ in H_0:

  - If $a^* = d^r$, then $x_i' = 0$. If $a^* = d^r h_i^t$, then $x_i' = 1$. This is equivalent to setting $x_i' = y \in \{0,1\}$ if $a^* = d^r h_i^{yt}$. Substituting in the above relations, this means that in H_0, $x_i' = y \in \{0,1\}$ if

$$a^* = d^r h_i^{yt} = \left(\prod_{j \in [N]} h_j^{x_j} \right)^r h_i^{yt} = h_i^{yt} (h_i^r)^{x_i} \prod_{j \neq i} (h_j^r)^{x_i}.$$

  - Otherwise $x_i' = \perp$.

  This is precisely the distribution of x_i' in H_1.

- Hybrids H_1 and H_2 are computationally indistinguishable under the DDH assumption and modeling H as a random oracle. To see this, suppose there exists an efficient adversary $\mathcal{A}$ that is able to distinguish hybrids H_1 and H_2 with non-negligible probability. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that distinguishes the distributions in Eq. (3.2):

  1. At the beginning of the game, algorithm $\mathcal{B}$ receives a challenge vector $(g, h_1, T_1, \ldots, h_N, T_N)$.
  2. Algorithm $\mathcal{B}$ programs the random oracle $H(i) \mapsto h_i$ for each $i \in [N]$. If $\mathcal{A}$ ever queries H on an input $k \notin [N]$, algorithm $\mathcal{B}$ samples a random $r_k \xleftarrow{\mathrm{R}} \mathbb{G}$ and defines the mapping $H(k) \mapsto r_k$.

3. Algorithm $\mathcal{B}$ now constructs the digest $d \leftarrow \prod_{j\in[N]} h_j^{x_j}$ as in H_1 and H_2. To construct the query, algorithm $\mathcal{B}$ sets $q_j \leftarrow T_j$ for $j \neq i$ and $q_i \leftarrow T_i h_i^t$ where $t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. It gives the digest d, the database x, and the query q to $\mathcal{A}$.
4. Algorithm $\mathcal{A}$ outputs a response a^*. Algorithm $\mathcal{B}$ computes x_i' as follows:
   - If $a^* = h_i^{yt} T_i^{x_i} \prod_{j\neq i} T_j^{x_j}$, then $x_i' \leftarrow x_i$.
   - Otherwise, $x_i' \leftarrow \perp$.
5. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with x_i' and outputs whatever $\mathcal{A}$ outputs.

Since $h_1, \ldots, h_N \xleftarrow{\mathrm{R}} \mathbb{G}$, the outputs of the random oracle are correctly simulated. Correspondingly, algorithm $\mathcal{B}$ perfectly simulates the distribution of the digest d for $\mathcal{A}$. We now consider the two possible challenge distributions:

- Suppose $T_i = h_i^r$ for all $i \in [N]$ and where $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. In this case, $q_j = h_j^r$ for all $j \neq i$ and $q_j = h_i^{r+t}$ where $t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. Similarly, $x_i' = y \in \{0,1\}$ if $a^* = h_i^{yt}(h_i^r)^{x_i} \prod_{j\neq i}(h_j^r)^{x_j}$, which exactly matches the distribution in H_1.
- Suppose $T_i = z_i \xleftarrow{\mathrm{R}} \mathbb{G}$ for all $i \in [N]$. In this case, $q_j = z_j$ for all $j \neq i$ and $q_j = z_i h_i^t$ where $t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. This is the query distribution in H_2. Similarly, to compute the response x_i', algorithm $\mathcal{B}$ sets $x_i' = y \in \{0,1\}$ if $a^* = h_i^{yt} z_i^{x_i} \prod_{j\neq i} z_j^{x_j}$, which matches the distribution in H_2.

We conclude that algorithm $\mathcal{B}$ distinguishes between the distributions in Eq. (3.2) with the same distinguishing advantage as $\mathcal{A}$, and the claim follows.

- Hybrids H_2 and H_3 are identically distributed. In H_2, the z_j's are sampled uniformly and independently from $\mathbb{G}$ (and also independent of $h_1, \ldots, h_N, t$). Thus, the distribution of $q = (q_1, \ldots, q_N)$ in H_2 is identical to that in H_3. Finally, in H_2, $q_i = z_i h_i^t$, where $t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. This is the distribution in H_3.

- The statistical distance between H_3 and H_4 is $1/p = \mathsf{negl}(\lambda)$. By construction, the two experiments are identical unless the adversary outputs a^* where $a^* = h_i^{(1-x_i)t} z_i^{x_i} \prod_{j\neq i} z_j^{x_j}$. Using the relation $z_i = q_i / h_i^t$, this becomes

$$a^* = h_i^{(1-x_i)t} \frac{q_i^{x_i}}{h_i^{x_i t}} \prod_{j\neq i} z_j^{x_j} = (h_i^t)^{1-2x_i} q_i^{x_i} \prod_{j\neq i} z_j^{x_j},$$

or equivalently, if

$$(h_i^t)^{1-2x_i} = \frac{a^*}{q_i^{x_i} \prod_{j\neq i} z_j^{x_j}}. \tag{B.6}$$

Now, in H_3 and H_4, the challenger samples $t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ *after* the adversary outputs a^*. Moreover, since $x_i \in \{0,1\}$, it follows that $1 - 2x_i \in \{-1, 1\}$. Since t is sampled independently of a^*, q_i and z_j for all $j \in [N]$, and h_i is a generator of $\mathbb{G}$ (with overwhelming probability), Eq. (B.6) holds with probability at most $1/p = \mathsf{negl}(\lambda)$ over the randomness of t.

- Hybrids H_4 and H_5 are identical experiments. In H_4, the challenger sets $x'_i = x_i$ if and only if
$$a^* = h_i^{x_i t} z_i^{x_i} \prod_{j \neq i} z_j^{x_j} = (z_i h_i^t)^{x_i} \prod_{j \neq i} z_j^{x_j} = \prod_{j \in [N]} q_j^{x_j},$$
since $q_j = z_j$ for all $j \neq i$ and $q_i = z_i h_i^t$. This is the distribution in H_5. □

Theorem 76 (Integrity of Construction 4)**.** Suppose the DDH assumption holds in $\mathbb{G}$ and H is modeled as a random oracle. Then, Construction 4 (instantiated with group $\mathbb{G}$ and hash function H) provides integrity.

Proof. Fix a database $x \in \{0,1\}^N$ and an index $i \in [N]$, and take any efficient adversary $\mathcal{A}$ for the integrity game. We define the following hybrid experiments:

- H_0: This is the real integrity game.
- H_1: Same as H_0, except the challenger samples $q \xleftarrow{R} \mathbb{G}^N$ and sets $x'_i \leftarrow x_i$ if $a^* = \prod_{j \in [N]} q_j^{x_j}$ and $x'_i \leftarrow \perp$ otherwise.

The outputs of H_0 and H_1 are computationally indistinguishable by Lemma 75. Next, in $H_1, \Pr[x'_i \notin \{x_i, \perp\}] = 0$ by construction. The claim now follows by a hybrid argument. □

Theorem 77 (Privacy of Construction 4)**.** Suppose the DDH assumption holds in $\mathbb{G}$ and H is modeled as a random oracle. Then, Construction 4 (instantiated with group $\mathbb{G}$ and hash function H) provides privacy.

Proof. Fix a database $x \in \{0,1\}^N$ and an index $i \in [N]$. Take any efficient adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. We construct an efficient simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ as follows:

Simulator $\mathcal{S}_0(1^\lambda, d, x)$	**Simulator** $\mathcal{S}_1(\mathsf{st}_\mathcal{S}, a^*)$
1 : $q = (q_1, \ldots, q_N) \xleftarrow{R} \mathbb{G}^N$	1 : $b \leftarrow \mathbb{1}\{a^* = \mathsf{st}_\mathcal{S}\}$
2 : $\mathsf{st}_\mathcal{S} \leftarrow \prod_{j \in [N]} q_j^{x_j}$	2 : **return** b
3 : **return** $(\mathsf{st}_\mathcal{S}, q)$	

We show that the real distribution $\text{REAL}_{\mathcal{A},x,i,\lambda}$ and ideal distribution $\text{IDEAL}_{\mathcal{A},\mathcal{S},x,\lambda}$ are computationally indistinguishable. We define a sequence of hybrid experiments:

- H_0: This is the real distribution $\text{REAL}_{\mathcal{A},x,i,\lambda}$.
- H_1: Same as H_0, except the challenger samples $q \xleftarrow{R} \mathbb{G}^N$ and sets $x'_i \leftarrow x_i$ if $a^* = \prod_{j \in [N]} q_j^{x_j}$ and $x'_i \leftarrow \perp$ otherwise.

- H_2: This is the ideal distribution $\text{IDEAL}_{\mathcal{A},\mathcal{S},x,\lambda}$.

We now argue that adjacent pair of hybrid experiments are indistinguishable:

- H_0 and H_1 are computationally indistinguishable by Lemma 75.
- H_1 and H_2 are identical experiments. Namely, in H_2, the challenger sets $b = 1$ if and only if $a^* = \prod_{j\in[N]} q_j^{x_j}$, which coincides with the behavior in H_1. □

B.4.2 Handling larger database rows

Our DDH-based construction (Construction 4) directly supports (small) multi-bit database records with no communication overhead. The cost is the client's computational cost increases by a factor of $2^{\ell/2}$, where ℓ is the bit-length of the record.

The idea is simple. Suppose the database consists of N ℓ-bit records $x_1, \ldots, x_\ell \in \{0,1\}^\ell$. The digest, query, and answer algorithms are unchanged (the only difference is that instead of each record $x_i \in \{0,1\}$ being a single bit, we now treat each record $x_i \in \{0,1\}^\ell$ as an integer between 0 and $2^\ell - 1$). The only difference is during reconstruction, the client now learns the value $h_i^{x_i t}$. Since the client knows the blinding factor t, it can exponentiate with $t^{-1} \bmod p$ to obtain $h_i^{x_i}$. Namely, the client is able to obtain an encoding of the database record in the exponent. Recovering the value of x_i now requires computing a discrete logarithm (base h_i). This can be computed in time $O(\sqrt{2^\ell})$ using Pollard's kangaroo method [Pol00], or alternatively, if ℓ is very small, then the client can precompute a lookup table of possible values for $h_i^{x_i}$. Thus, this approach is suitable for small values of ℓ (e.g., $\ell \leq 32$).

While there are applications for a small-row single-server authenticated PIR scheme, we still hope that it is possible to construct a more bandwidth- and computation-efficient scheme in the future. We unsuccessfully attempted to combine an unauthenticated classic single-server PIR scheme with some sort of algebraic integrity-protection mechanism, but it seems non-trivial to provide our integrity properties while making only black-box use of the underlying single-server PIR scheme. Further investigation along these lines would be an interesting task for future work.

Supporting multi-bit records in the lattice-based setting. We note that a similar approach as above can be applied to the lattice-based construction (Construction 3) to support multi-bit records. While correctness holds, the security analysis is more challenging. Namely, both integrity and privacy of Construction 3 (Theorems 73 and 74) rely on the extended LWE assumption where we require that LWE holds even if the distinguisher is given a linear combination $\mathbf{e}^\intercal\mathbf{x}$ of the LWE error. When the database entries are binary-valued (i.e., $\mathbf{x} \in \{0,1\}^N$), we can appeal to [BLP+13, Lemma 4.3,

Claim 4.6, Lemma 4.7] to base hardness on standard LWE. It seems plausible that a similar (possibly less tight) reduction applies when the database $\mathbf{x} \in (\{0,1\}^\ell)^N$ consists of ℓ-bit integers, and this is an interesting question for further exploration.

Bibliography

[AAB+24] Harold Abelson, Ross J. Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Vanessa Teague, and Carmela Troncoso. Bugs in our pockets: the risks of client-side scanning. *J. Cybersecur.*, 2024. (Cited on page 6.)

[AAL+05] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. DNS Security Introduction and Requirements. RFC 4003, 2005. (Cited on page 64.)

[ABJM21] Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. Collective Information Security in Large-Scale Urban Protests: the Case of Hong Kong. In *USENIX Security*, 2021. (Cited on pages 2 and 6.)

[ACD18] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol. Cryptology ePrint Archive, 2018. (Cited on pages 3 and 15.)

[ACD19] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol. In *EUROCRYPT*, 2019. (Cited on pages 3, 4, 14, 15, 16, 18, 19, 20, 21, 25, 33, 40, and 54.)

[ACDT21] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Modular Design of Secure Group Messaging Protocols and the Security of MLS. In *CCS*, 2021. (Cited on page 54.)

[ACJM20] Joël Alwen, Sandro Coretti, Daniel Jost, and Marta Mularczyk. Continuous group key agreement with active security. In *TCC*, 2020. (Cited on page 54.)

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with Compressed Queries and Amortized Query Processing. In *IEEE S&P*, 2018. (Cited on pages 8, 58, 60, and 90.)

[AJ20] Martin R. Albrecht and Rikke Bjerg Jensen. The Vacuity of the Open Source Security Testing Methodology Manual. In *SSR*, 2020. (Cited on page 6.)

[AJM22] Joël Alwen, Daniel Jost, and Marta Mularczyk. On the Insider Security of MLS. In *CRYPTO*, 2022. (Cited on page 54.)

[Ajt96] Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *STOC*, 1996. (Cited on page 92.)

[ALP+21] Asra Ali, Tancrede Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–Computation Trade-offs in PIR. In *USENIX Security*, 2021. (Cited on pages 8, 58, and 60.)

[AMBFK16] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private Information Retrieval for Everyone. *PoPETs*, 2016. (Cited on pages 8, 58, and 60.)

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *J. Math. Cryptol.*, 2015. (Cited on page 98.)

[AQ22] Marcel Armour and Elizabeth Quaglia. Subverting Deniability. In *ProvSec*, 2022. (Cited on page 117.)

[AS16] Sebastian Angel and Srinath T. V. Setty. Unobservable Communication over Fully Untrusted Infrastructure. In *OSDI*, 2016. (Cited on pages 8, 58, and 68.)

[BCC+23a] Khashayar Barooti, Daniel Collins, Simone Colombo, Loïs Huguenin-Dumittan, and Serge Vaudenay. On Active Attack Detection in Messaging with Immediate Decryption. In *CRYPTO*, 2023. (Cited on pages 8, 11, 13, 16, 17, 40, and 55.)

[BCC+23b] Khashayar Barooti, Daniel Collins, Simone Colombo, Loïs Huguenin-Dumittan, and Serge Vaudenay. On Active Attack Detection in Messaging with Immediate Decryption. Cryptology ePrint Archive, 2023. (Cited on pages 11, 13, and 48.)

[BCG+21] Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function Secret Sharing for Mixed-Mode and Fixed-Point Secure Computation. In *EUROCRYPT*, 2021. (Cited on pages 86 and 106.)

[BDK11] Lars Backstrom, Cynthia Dwork, and Jon M. Kleinberg. Wherefore art thou R3579X?: anonymized social networks, hidden patterns, and structural steganography. *Commun. ACM*, 2011. (Cited on page 5.)

[BDKP22] Shany Ben-David, Yael Tauman Kalai, and Omer Paneth. Verifiable Private Information Retrieval. In *TCC*, 2022. (Cited on page 106.)

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic Encryption and Multiparty Computation. In *EUROCRYPT*, 2011. (Cited on page 106.)

[BE24] Connor Bell and Saba Eskandarian. Anonymous Complaint Aggregation for Secure Messaging. *PoPETs*, 2024. (Cited on page 115.)

[BFG+22a] Alexander Bienstock, Jaiden Fairoze, Sanjam Garg, Pratyay Mukherjee, and Srinivasan Raghuraman. A More Complete Analysis of the Signal Double Ratchet Algorithm. In *CRYPTO*, 2022. (Cited on page 19.)

[BFG+22b] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake. In *PKC*, 2022. (Cited on pages 108, 111, and 112.)

[BG21] Colin Boyd and Kai Gellert. A Modern View on Forward Security. *Comput. J.*, 2021. (Cited on page 4.)

[BGB04] Nikita Borisov, Ian Goldberg, and Eric A. Brewer. Off-the-record communication, or, why not to use PGP. In *WPES*, 2004. (Cited on pages 4, 10, 108, and 111.)

[BGG94] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental Cryptography: The Case of Hashing and Signing. In *CRYPTO*, 1994. (Cited on page 42.)

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function Secret Sharing. In *EUROCRYPT*, 2015. (Cited on pages 9, 60, 85, 86, 99, 100, 101, and 106.)

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *CCS*, 2016. (Cited on pages 8, 9, 58, 60, 76, 85, 86, 89, 99, 100, 105, and 106.)

[BI01] Amos Beimel and Yuval Ishai. Information-Theoretic Private Information Retrieval: A Unified Construction. In *ICALP*, 2001. (Cited on page 105.)

[BIKR02] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-François Raymond. Breaking the $\mathcal{O}(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval. In *FOCS*, 2002. (Cited on page 105.)

[BIM04] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing. *J. Cryptol.*, 2004. (Cited on page 101.)

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can We Access a Database Both Locally and Privately? In *TCC*, 2017. (Cited on pages 8 and 58.)

[BJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key Agreement Protocols and Their Security Analysis. In *IMACC*, 1997. (Cited on page 4.)

[Bla12] Jean-François Blanchette. *Burdens of Proof: Cryptographic Culture and Evidence Law in the Age of Electronic Documents*. MIT Press, 2012. (Cited on page 6.)

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013. (Cited on pages 91, 92, and 168.)

[Bon98] Dan Boneh. The Decision Diffie-Hellman Problem. In *ANTS*, 1998. (Cited on page 9.)

[BP04] Mihir Bellare and Adriana Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *CRYPTO*, 2004. (Cited on page 111.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, 1993. (Cited on pages 158 and 163.)

[Bre21] Vincent Breitmoser. Private communication, 2021. (Cited on page 104.)

[BRV20] Fatih Balli, Paul Rösler, and Serge Vaudenay. Determining the Core Primitive for Optimally Secure Ratcheting. In *ASIACRYPT*, 2020. (Cited on page 54.)

[BS02] Amos Beimel and Yoav Stahl. Robust Information-Theoretic Private Information Retrieval. In *SCN*, 2002. (Cited on pages 9, 58, 62, 68, and 105.)

[BS07] Amos Beimel and Yoav Stahl. Robust Information-Theoretic Private Information Retrieval. *J. Cryptol.*, 2007. (Cited on pages 9, 58, 62, 68, and 105.)

[BSJ+15] Richard L. Barnes, Bruce Schneier, Cullen Jennings, Ted Hardie, Brian Trammell, Christian Huitema, and Daniel Borkmann. Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement. *RFC*, 7624, 2015. (Cited on page 4.)

[BSJ+17] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. Ratcheted Encryption and Key Exchange: The Security of Messaging. In *CRYPTO*, 2017. (Cited on pages 4, 14, 25, and 54.)

[BSSW02] Dirk Balfanz, Diana K. Smetters, Paul Stewart, and H. Chi Wong. Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In *NDSS*, 2002. (Cited on page 48.)

[Cap13] Justin Cappos. Avoiding theoretical optimality to efficiently and privately retrieve security updates. In *FC*, 2013. (Cited on pages 8 and 58.)

[Cas20] Ryan Castellucci. DKIM: Show Your Privates. https://rya.nc/dkim-privates.html, 2020. Last visited on 26-10-2023. (Cited on page 128.)

[CCG16] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. On Post-compromise Security. In *CSF*, 2016. (Cited on pages 4 and 14.)

[CCHD23] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real-World Deniability in Messaging. Cryptology ePrint Archive, 2023. (Cited on pages 11, 107, and 131.)

[CCHD25] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real-World Deniability in Messaging. *PoPETs*, 2025. (Cited on pages 11 and 107.)

[CDDF20] Sébastien Campion, Julien Devigne, Céline Duguey, and Pierre-Alain Fouque. Multi-Device for Signal. In *ACNS*, 2020. (Cited on page 5.)

[CDF+08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors. In *EUROCRYPT*, 2008. (Cited on pages 9, 60, 86, 89, and 106.)

[CDG+22] Brian Chen, Yevgeniy Dodis, Esha Ghosh, Eli Goldin, Balachandar Kesavan, Antonio Marcedone, and Merry Ember Mou. Rotatable Zero Knowledge Sets: Post Compromise Secure Auditable Dictionaries with application to Key Transparency. In *ASIACRYPT*, 2022. (Cited on pages 5 and 106.)

[CDGM19] Melissa Chase, Apoorvaa Deshpande, Esha Ghosh, and Harjasleen Malvai. SEEMless: Secure End-to-End Encrypted Messaging with less Trust. In *CCS*, 2019. (Cited on pages 5, 54, and 106.)

[CDV21] Andrea Caforio, F. Betül Durak, and Serge Vaudenay. Beyond Security and Efficiency: On-Demand Ratcheting with Security Awareness. In *PKC*, 2021. (Cited on pages 7, 13, 14, 15, 16, 19, 24, 38, and 54.)

[CDvD+03] Dwaine E. Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend, and G. Edward Suh. Incremental Multiset Hash Functions and Their Application to Memory Integrity Checking. In *ASIACRYPT*, 2003. (Cited on page 42.)

[CF11] Cas Cremers and Michèle Feltz. One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability. Cryptology ePrint Archive, 2011. (Cited on page 111.)

[CFKN20] Cas Cremers, Jaiden Fairoze, Benjamin Kiesl, and Aurora Naska. Clone Detection in Secure Messaging: Improving Post-Compromise Security in Practice. In *CCS*, 2020. (Cited on page 5.)

[CG97] Benny Chor and Niv Gilboa. Computationally Private Information Retrieval (Extended Abstract). In *STOC*, 1997. (Cited on page 105.)

[CGCD+17] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *EuroS&P*, 2017. (Cited on page 14.)

[CGG+22] Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, Jovan Komatovic, Zarko Milosevic, and Adi Seredinschi. Crime and Punishment in Distributed Byzantine Decision tasks. In *ICDCS*, 2022. (Cited on page 54.)

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *FOCS*, 1995. (Cited on pages 3, 8, 58, 62, 63, 76, 90, and 105.)

[CGN98] Benny Chor, Nic Gilboa, and Mori Naor. Private Information Retrieval by Keywords. Cryptology ePrint Archive, 1998. (Cited on page 63.)

[CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-Abuse Attacks Against Searchable Encryption. In *CCS*, 2015. (Cited on page 121.)

[Cha18] Luke Champine. fastxor. https://github.com/lukechampine/fastxor, 2018. (Cited on page 97.)

[Cha22] Luke Champine. uint128 for Go. https://github.com/lukechampine/uint128, 2022. (Cited on page 96.)

[CHK11] D. Crocker, T. Hansen, and M. Kucherawy. Domainkeys Identified Mail (DKIM) Signatures. STD 76, RFC Editor, September 2011. http://www.rfc-editor.org/rfc/rfc6376.txt. (Cited on pages 10, 109, 126, and 127.)

[CHMR23] Suvradip Chakraborty, Dennis Hofheinz, Ueli Maurer, and Guilherme Rito. Deniable Authentication When Signing Keys Leak. In *EUROCRYPT*, 2023. (Cited on page 108.)

[CJN23] Cas Cremers, Charlie Jacomme, and Aurora Naska. Formal Analysis of Session-Handling in Secure Messaging: Lifting Security from Sessions to Conversations. In *USENIX Security*, 2023. (Cited on page 5.)

[CK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private Information Retrieval with Sublinear Online Time. In *EUROCRYPT*, 2020. (Cited on pages 8, 58, and 76.)

[CMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single Database Private Information Retrieval Implies Oblivious Transfer. In *EUROCRYPT*, 2000. (Cited on pages 62 and 63.)

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *EUROCRYPT*, 1999. (Cited on pages 8, 58, and 105.)

[CN20] Aloni Cohen and Kobbi Nissim. Towards formalizing the GDPR's notion of singling out. *Proc. Natl. Acad. Sci. USA*, 2020. (Cited on page 115.)

[CN21] Justin D. Cochran and Stuart Napshin. Deepfakes: Awareness, Concerns, and Platform Accountability. *Cyberpsychology Behav. Soc. Netw.*, 2021. (Cited on page 134.)

[CNCG+23a] Simone Colombo, Kirill Nikitin, Henry Corrigan-Gibbs, David J. Wu, and Bryan Ford. Authenticated private information retrieval. In *USENIX Security*, 2023. (Cited on pages 8, 11, and 57.)

[CNCG+23b] Simone Colombo, Kirill Nikitin, Henry Corrigan-Gibbs, David J. Wu, and Bryan Ford. Authenticated private information retrieval. Cryptology ePrint Archive, 2023. (Cited on pages 11 and 57.)

[CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable Adaptive Oblivious Transfer. In *EUROCRYPT*, 2007. (Cited on page 72.)

[Col24] Daniel Patrick Collins. *On the Theory and Practice of Modern Secure Messaging*. PhD thesis, EPFL, 2024. (Cited on page 13.)

[CS20] Sofía Celi and Iraklis Symeonidis. The current state of denial. In *HotPETs*, 2020. (Cited on page 115.)

[CZ22] Cas Cremers and Mang Zhao. Provably Post-Quantum Secure Messaging with Strong Compromise Resilience and Immediate Decryption. Cryptology ePrint Archive, 2022. (Cited on page 14.)

[dCP22] Leo de Castro and Antigoni Polychroniadou. Lightweight, Maliciously Secure Verifiable Function Secret Sharing. In *EUROCRYPT*, 2022. (Cited on pages 86 and 106.)

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography (Extended Abstract). In *STOC*, 1991. (Cited on page 64.)

[DFD+21] Emma Dauterman, Vivian Fang, Ioannis Demertzis, Natacha Crooks, and Raluca Ada Popa. Snoopy: Surpassing the Scalability Bottleneck of Oblivious Storage. In *SOSP*, 2021. (Cited on page 5.)

[DFG+13] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A Cryptographic Analysis of OPACITY - (Extended Abstract). In *ESORICS*, 2013. (Cited on page 111.)

[DFL+20] Emma Dauterman, Eric Feng, Ellen Luo, Raluca Ada Popa, and Ion Stoica. DORY: An Encrypted Search System with Distributed Trust. In *OSDI*, 2020. (Cited on page 106.)

[DG16] Zeev Dvir and Sivakanth Gopi. 2-Server PIR with subpolynomial communication. *J. ACM*, 2016. (Cited on page 105.)

[DGH12] Casey Devet, Ian Goldberg, and Nadia Heninger. Optimally Robust Private Information Retrieval. In *USENIX Security*, 2012. (Cited on pages 9, 58, 62, and 106.)

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor Hash Functions and Their Applications. In *CRYPTO*, 2019. (Cited on page 105.)

[DGP22] Benjamin Dowling, Felix Günther, and Alexandre Poirrier. Continuous Authentication in Secure Messaging. In *ESORICS*, 2022. (Cited on pages 7, 14, 15, 24, 45, 51, and 54.)

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 1976. (Cited on page 4.)

[DH20] Benjamin Dowling and Britta Hale. There Can Be No Compromise: The Necessity of Ratcheted Authentication in Secure Messaging. Cryptology ePrint Archive, 2020. (Cited on pages 14 and 24.)

[DH21] Benjamin Dowling and Britta Hale. Secure Messaging Authentication against Active Man-in-the-Middle Attacks. In *EuroS&P*, 2021. (Cited on pages 7, 14, and 15.)

[DHRR22] Benjamin Dowling, Eduard Hauck, Doreen Riepel, and Paul Rösler. Strongly Anonymous Ratcheted Key Exchange. In *ASIACRYPT*, 2022. (Cited on page 54.)

[DKSW09] Yevgeniy Dodis, Jonathan Katz, Adam D. Smith, and Shabsi Walfish. Composability and On-Line Deniability of Authentication. In *TCC*, 2009. (Cited on pages 108 and 111.)

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent Zero-Knowledge. In *STOC*, 1998. (Cited on pages 4 and 111.)

[DPC23] Alex Davidson, Gonçalo Pestana, and Sofía Celi. FrodoPIR: Simple, Scalable, Single-Server Private Information Retrieval. In *PoPETs*, 2023. (Cited on pages 8, 58, and 90.)

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *CRYPTO*, 2012. (Cited on pages 9, 60, 86, and 106.)

[Dro24] Dropwizard. Dropwizard. https://github.com/dropwizard/dropwizard, 2024. (Cited on page 122.)

[DSKB21] Alaa Daffalla, Lucy Simko, Tadayoshi Kohno, and Alexandru G. Bardas. Defensive Technology Use by Political Activists During the Sudanese Revolution. In *IEEE S&P*, 2021. (Cited on pages 6 and 136.)

[DT24] Marian Dietz and Stefano Tessaro. Fully Malicious Authenticated PIR. In *CRYPTO*, 2024. (Cited on pages 69 and 143.)

[DV19] F. Betül Durak and Serge Vaudenay. Bidirectional Asynchronous Ratcheted Key Agreement with Linear Complexity. In *IWSEC*, 2019. (Cited on pages 7, 13, 14, 15, 16, and 54.)

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and Authenticated Key Exchanges. *Des. Codes Cryptogr.*, 1992. (Cited on page 4.)

[EGS21] Peter Elkind, Jack Gillum, and Craig Silverman. How Facebook Undermines Privacy Protections for Its 2 Billion WhatsApp Users. https://www.propublica.org/article/how-facebook-undermines-privacy-protections-for-its-2-billion-whatsapp-users, 2021. Last visited on 26-10-2023. (Cited on page 118.)

[EKN22] Reo Eriguchi, Kaoru Kurosawa, and Koji Nuida. Multi-Server PIR with Full Error Detection and Limited Error Correction. In *ITC*, 2022. (Cited on page 106.)

[EM19] Ksenia Ermoshina and Francesca Musiani. "standardising by running code": the Signal protocol and de facto standardisation in end-to-end encrypted messaging. *Internet Histories*, 2019. (Cited on page 4.)

[FHK19] Armando Faz-Hernández and Kris Kwiatkowski. Introducing CIRCL: An Advanced Cryptographic Library. https://github.com/cloudflare/circl, 2019. (Cited on page 96.)

[FJ24] Rune Fiedler and Christian Janson. A Deniability Analysis of Signal's Initial Handshake PQXDH. *PoPETs*, 2024. (Cited on pages 111, 112, and 137.)

[FM15] Marc Fischlin and Sogol Mazaheri. Notions of deniable message authentication. In *WPES*, 2015. (Cited on page 111.)

[FPS+18] Jonathan Frankle, Sunoo Park, Daniel Shaar, Shafi Goldwasser, and Daniel J. Weitzner. Practical Accountability of Secret Processes. In *USENIX Security*, 2018. (Cited on pages 6 and 115.)

[FS09] Marc Fischlin and Dominique Schröder. Security of Blind Signatures under Aborts. In *PKC*, 2009. (Cited on page 72.)

[GCM+16] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath T. V. Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and Private Media Consumption with Popcorn. In *NSDI*, 2016. (Cited on pages 8, 58, 59, and 63.)

[GGV20] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. Formalizing Data Deletion in the Context of the Right to Be Forgotten. In *EUROCRYPT*, 2020. (Cited on pages 6 and 115.)

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with Applications to PIR. In *TCC*, 2019. (Cited on pages 8 and 58.)

[GI14] Niv Gilboa and Yuval Ishai. Distributed Point Functions and Their Applications. In *EUROCRYPT*, 2014. (Cited on pages 63, 76, 99, and 100.)

[GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message Franking via Committing Authenticated Encryption. In *CRYPTO*, 2017. (Cited on page 115.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, 1987. (Cited on page 105.)

[Gol07] Ian Goldberg. Improving the Robustness of Private Information Retrieval. In *IEEE S&P*, 2007. (Cited on pages 9, 58, 62, 68, and 106.)

[GPA19] Lachlan J. Gunn, Ricardo Vieitez Parra, and N. Asokan. Circumventing Cryptographic Deniability with Remote Attestation. *PoPETs*, 2019. (Cited on page 117.)

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *ICALP*, 2005. (Cited on page 105.)

[Gre14a] Matthew Green. Noodling about IM protocols. https://blog.cryptographyengineering.com/2014/07/26/noodling-about-im-protocols/, 2014. Last visited on 02-11-2022. (Cited on pages 10 and 108.)

[Gre14b] Glenn Greenwald. *No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State*. Metropolitan Books, 2014. (Cited on page 1.)

[GRS17] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. Why Your Encrypted Database Is Not Secure. In *HotOS*, 2017. (Cited on page 121.)

[GSB+17] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-Abuse Attacks against Order-Revealing Encryption. In *IEEE S&P*, 2017. (Cited on page 121.)

[Gua13] The Guardian. The NSA Files, 2013. Last visited on 2024-09-10. (Cited on page 1.)

[Gün89] Christoph G. Günther. An Identity-Based Key-Exchange Protocol. In *EUROCRYPT*, 1989. (Cited on page 4.)

[HDCZ23] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private Web Search with Tiptoe. In *SOSP*, 2023. (Cited on page 58.)

[Hea14] Mike Hearn. Value of deniability (mailing list discussion). https://moderncrypto.org/mail-archive/messaging/2014/001173.html, 2014. Last visited on 09-11-2022. (Cited on page 111.)

[HHC+23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. In *USENIX Security*, 2023. (Cited on page 58.)

[HHCG+23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. In *USENIX Security*, 2023. (Cited on pages 8, 58, 60, 90, 92, 93, 97, 101, 102, and 104.)

[HKD07] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: Practical accountability for distributed systems. *SIGOPS*, 2007. (Cited on page 54.)

[HKE13] Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *CRYPTO*, 2013. (Cited on pages 9, 59, and 105.)

[HKKP22] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for signal's handshake (X3DH): post-quantum, state leakage secure, and deniable. *Journal of Cryptology*, 2022. (Cited on page 112.)

[HLLC11] Lein Harn, Chia-Yin Lee, Changlu Lin, and Chin-Chen Chang. Fully Deniable Message Authentication Protocols Preserving Confidentiality. *Comput. J.*, 2011. (Cited on page 111.)

[HSW23] Laura Hetz, Thomas Schneider, and Christian Weinert. Scaling Mobile Private Contact Discovery to Billions of Users. In *ESORICS*, 2023. (Cited on page 5.)

[HW21] Andreas Hülsing and Florian Weber. Epochal Signatures for Deniable Group Chats. In *IEEE S&P*, 2021. (Cited on page 111.)

[HWS+21] Christoph Hagen, Christian Weinert, Christoph Sendner, Alexandra Dmitrienko, and Thomas Schneider. All the Numbers are US: Large-scale Abuse of Contact Discovery in Mobile Messengers. In *NDSS*, 2021. (Cited on page 5.)

[IAV22] Rawane Issa, Nicolas Alhaddad, and Mayank Varia. Hecate: Abuse Reporting in Secure Messengers with Sealed Sender. In *USENIX Security*, 2022. (Cited on page 115.)

[IOZ14] Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *CRYPTO*, 2014. (Cited on page 54.)

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *TCC*, 2009. (Cited on page 72.)

[JMM19] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient Ratcheting: Almost-Optimal Guarantees for Secure Messaging. In *EUROCRYPT*, 2019. (Cited on pages 19 and 54.)

[JS18] Joseph Jaeger and Igors Stepanovs. Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging. In *CRYPTO*, 2018. (Cited on page 54.)

[KC21] Dmitry Kogan and Henry Corrigan-Gibbs. Private Blocklist Lookups with Checklist. In *USENIX Security*, 2021. (Cited on page 58.)

[KFR09] Ronald Kainda, Ivan Flechais, and A. W. Roscoe. Usability and security of out-of-band channels in secure device pairing protocols. In *SOUPS*, 2009. (Cited on pages 5, 7, and 15.)

[KLDF] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle: An Efficient Communication System With Strong Anonymity. *PoPETs*, 2016. (Cited on page 68.)

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval. In *FOCS*, 1997. (Cited on pages 8, 9, 58, 59, 60, 61, 62, 63, 105, and 106.)

[Kre23] Ehren Kret. Quantum Resistance and the Signal Protocol. https://signal.org/blog/pqxdh/, 2023. Last visited on 26-10-2023. (Cited on page 111.)

[KRS+19] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In *USENIX Security*, 2019. (Cited on page 5.)

[KS06] Mehmet S. Kiraz and Berry Schoenmakers. A Protocol Issue for the Malicious Case of Yao's Garbled Circuit Construction. In *SITB*, 2006. (Cited on pages 9, 59, 62, 64, and 105.)

[KS23] Ehren Kret and Rolfe Schmidt. The PQXDH Key Agreement Protocol. https://signal.org/docs/specifications/pqxdh/, 2023. Last visited on 23-11-2023. (Cited on pages 111 and 137.)

[Kur19] Kaoru Kurosawa. How to Correct Errors in Multi-server PIR. In *ASIACRYPT*, 2019. (Cited on pages 62 and 106.)

[Lau14] Ben Laurie. Certificate transparency. *Commun. ACM*, 2014. (Cited on pages 5 and 106.)

[LG15] Wouter Lueks and Ian Goldberg. Sublinear Scaling for Multi-Client Private Information Retrieval. In *FC*, 2015. (Cited on pages 8 and 58.)

[Lip05] Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In *ISC*, 2005. (Cited on pages 8, 58, and 105.)

[LP11] Yehuda Lindell and Benny Pinkas. Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. In *TCC*, 2011. (Cited on page 105.)

[LPA+19] Lucy Li, Bijeeta Pal, Junade Ali, Nick Sullivan, Rahul Chatterjee, and Thomas Ristenpart. Protocols for checking compromised credentials. In *CCS*, 2019. (Cited on pages 8 and 58.)

[Lun18] Joshua Lund. Technology preview: Sealed sender for Signal. https://signal.org/blog/sealed-sender/, 2018. Last visited on 03-11-2022. (Cited on pages 5, 109, and 122.)

[Mar13] Moxie Marlinspike. Simplifying OTR deniability. https://signal.org/blog/simplifying-otr-deniability/, 2013. Last visited on 25-10-2022. (Cited on pages 10 and 108.)

[Mar17a] Moxie Marlinspike. Private Contact Discovery. https://signal.org/blog/private-contact-discovery/, September 26, 2017. Last visited on 11-04-2021. (Cited on pages 58 and 97.)

[Mar17b] Moxie Marlinspike. Safety number updates. https://signal.org/blog/verified-safety-number-updates/, 2017. Last visited on 22-05-2022. (Cited on pages 5, 14, 16, and 51.)

[MBB+15] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: Bringing Key Transparency to End Users. In *USENIX Security*, 2015. (Cited on pages 5, 54, 59, 63, 69, 90, 97, and 106.)

[MCR21] Muhammad Haris Mughees, Hao Chen, and Ling Ren. OnionPIR: Response Efficient Single-Server PIR. In *CCS*, 2021. (Cited on pages 8, 58, 90, and 101.)

[MCYR17] Kevin Milner, Cas Cremers, Jiangshan Yu, and Mark Ryan. Automatically Detecting the Misuse of Secrets: Foundations, Design Principles, and Applications. In *CSF*, 2017. (Cited on page 54.)

[Mer87] Ralph C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *CRYPTO*, 1987. (Cited on page 73.)

[mit] MIT PGP Public Key Server. https://pgp.mit.edu/. Last visited on 9-04-2021. (Cited on page 97.)

[MKA+21] Ian Martiny, Gabriel Kaptchuk, Adam J Aviv, Daniel S Roche, and Eric Wustrow. Improving Signal's Sealed Sender. In *NDSS*, 2021. (Cited on page 116.)

[MKS+23] Harjasleen Malvai, Lefteris Kokoris-Kogias, Alberto Sonnino, Esha Ghosh, Ercan Oztürk, Kevin Lewi, and Sean F. Lawlor. Parakeet: Practical Key Transparency for End-to-End Encrypted Messaging. In *NDSS*, 2023. (Cited on page 5.)

[MOT+11] Prateek Mittal, Femi G Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. In *USENIX Security*, 2011. (Cited on page 106.)

[MP16] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. https://www.signal.org/docs/specifications/x3dh/, 2016. Last visited on 25-10-2022. (Cited on pages 4, 10, 108, 111, 122, and 137.)

[MP17] Moxie Marlinspike and Trevor Perrin. The Sesame Algorithm: Session Management for Asynchronous Message Encryption. https://signal.org/docs/specifications/sesame/, 2017. Last visited on 25-10-2022. (Cited on page 5.)

[MPC+18] Pratyush Mishra, Rishabh Poddar, Jerry Chen, Alessandro Chiesa, and Raluca Ada Popa. Oblix: An efficient oblivious search index. In *IEEE S&P*, 2018. (Cited on pages 5 and 106.)

[MSGJ24] Nicolas Mohnblatt, Alberto Sonnino, Kobi Gurkan, and Philipp Jovanovic. Arke: Scalable and Byzantine Fault Tolerant Privacy-Preserving Contact Discovery. *CCS*, 2024. (Cited on page 5.)

[MW22] Samir Jordan Menon and David J. Wu. SPIRAL: Fast, High-Rate Single-Server PIR via FHE Composition. In *IEEE S&P*, 2022. (Cited on pages 8, 58, 90, and 101.)

[Nik21] Kirill Nikitin. *Integrity and Metadata Protection in Data Retrieval*. PhD thesis, EPFL, 2021. (Cited on page 57.)

[NKKJ+17] Kirill Nikitin, Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Justin Cappos, and Bryan Ford. CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds. In *USENIX Security*, 2017. (Cited on page 59.)

[NR97] Moni Naor and Omer Reingold. Number-theoretic Constructions of Efficient Pseudo-random Functions. In *FOCS*, 1997. (Cited on page 93.)

[NRS20] Moni Naor, Lior Rotem, and Gil Segev. Out-Of-Band Authenticated Group Key Exchange: From Strong Authentication to Immediate Key Delivery. In *ITC*, 2020. (Cited on page 54.)

[NS09] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing Social Networks. In *IEEE S&P*, 2009. (Cited on page 5.)

[NSR11] Arvind Narayanan, Elaine Shi, and Benjamin I. P. Rubinstein. Link prediction by de-anonymization: How We Won the Kaggle Social Network Challenge. In *IJCNN*, 2011. (Cited on page 5.)

[OG10] Femi G. Olumofin and Ian Goldberg. Privacy-Preserving Queries over Relational Databases. In *PoPETs*, 2010. (Cited on pages 8, 58, and 105.)

[OS07] Rafail Ostrovsky and William E Skeith. A survey of single-database private information retrieval: Techniques and applications. In *PKC*, 2007. (Cited on pages 9, 61, and 105.)

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, 1991. (Cited on page 94.)

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *IEEE S&P*, 2013. (Cited on page 90.)

[PIB+22] Bijeeta Pal, Mazharul Islam, Marina Sanusi Bohuk, Nick Sullivan, Luke Valenta, Tara Whalen, Christopher A. Wood, Thomas Ristenpart, and Rahul Chatterjee. Might I Get Pwned: A Second Generation Compromised Credential Checking Service. In *USENIX Security*, 2022. (Cited on page 58.)

[PM16] Trevor Perrin and Moxie Marlinspike. The Double Ratchet Algorithm. https://signal.org/docs/specifications/doubleratchet/, 2016. Last visited on 25-10-2022. (Cited on pages 4, 10, 14, 18, 20, 42, 108, 111, 122, and 142.)

[Pol00] John M. Pollard. Kangaroos, Monopoly and Discrete Logarithms. *J. Cryptol.*, 2000. (Cited on page 167.)

[PP22] Jeroen Pijnenburg and Bertram Poettering. On Secure Ratcheting with Immediate Decryption. In *ASIACRYPT*, 2022. (Cited on pages 14 and 54.)

[PPY18] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private Stateful Information Retrieval. In *CCS*, 2018. (Cited on pages 8 and 58.)

[PR18a] Bertram Poettering and Paul Rösler. Asynchronous ratcheted key exchange. Cryptology ePrint Archive, 2018. (Cited on page 14.)

[PR18b] Bertram Poettering and Paul Rösler. Towards Bidirectional Ratcheted Key Exchange. In *CRYPTO*, 2018. (Cited on page 54.)

[PT20] Jeongeun Park and Mehdi Tibouchi. SHECS-PIR: Somewhat Homomorphic Encryption-Based Compact and Scalable Private Information Retrieval. In *ESORICS*, 2020. (Cited on pages 8 and 58.)

[PV06] Sylvain Pasini and Serge Vaudenay. An Optimal Non-interactive Message Authentication Protocol. In *CT-RSA*, 2006. (Cited on pages 38 and 51.)

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005. (Cited on pages 9, 91, and 92.)

[RG05] Mario Di Raimondo and Rosario Gennaro. New approaches for deniable authentication. In *CCS*, 2005. (Cited on pages 108 and 111.)

[RGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In *CCS*, 2006. (Cited on pages 10, 108, 109, and 111.)

[RMA+23] N. Reitinger, N. Malkin, O. Akgul, M. L. Mazurek, and I. Miers. Is Cryptographic Deniability Sufficient? Non-Expert Perceptions of Deniability in Secure Messaging. In *IEEE S&P*, 2023. (Cited on pages 6, 110, 112, 113, 138, and 144.)

[Rog06] Phillip Rogaway. Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys. Cryptology ePrint Archive, 2006. (Cited on page 28.)

[Rog15] Phillip Rogaway. The Moral Character of Cryptographic Work. Cryptology ePrint Archive, 2015. (Cited on pages 1, 141, and 143.)

[RPG07] Joel Reardon, Jeffrey Pound, and Ian Goldberg. Relational- Complete Private Information Retrieval. *Technical Report CACR*, 2007. (Cited on page 105.)

[Rya14] Mark Dermot Ryan. Enhanced Certificate Transparency and End-to-End Encrypted Mail. In *NDSS*, 2014. (Cited on pages 5, 8, 58, and 106.)

[RYAJ+24] Anamika Rajendran, Tarun Kumar Yadav, Malek Al-Jbour, Francisco Manuel Mares Solano, Kent Seamons, and Joshua Reynolds. Deniable Encrypted Messaging: User Understanding after Hands-on Social Experience. In *EuroUSEC*, 2024. (Cited on page 144.)

[Sig21a] Signal. Grand jury subpoena for Signal user data, Central District of California. https://signal.org/bigbrother/central-california-grand-jury/, 2021. Last visited on 25-10-2022. (Cited on page 110.)

[Sig21b] Signal. Grand jury subpoena for Signal user data, Central District of California (again!). https://signal.org/bigbrother/cd-california-grand-jury/, 2021. Last visited on 25-10-2022. (Cited on page 110.)

[Sig22] Signal. Signal-Server. https://github.com/signalapp/Signal-Server, 2022. (Cited on page 122.)

[SMF24] Matan Shtepel, Pratyush Mishra, and Brett Falk. Private communication, 2024. (Cited on page 78.)

[SPG21] Michael A. Specter, Sunoo Park, and Matthew Green. KeyForge: Non-Attributable Email from Forward-Forgeable Signatures. In *USENIX Security*, 2021. (Cited on pages 10, 107, 109, 110, 122, 126, and 127.)

[SRCM+22] John Scott-Railton, Elies Campo, Bill Marczak, Bahr Abdul Razzak, Siena Anstis, Gözde Böcü, Salvatore Solimano, and Ron Deibert. CatalanGate: Extensive Mercenary Spyware Operation against Catalans Using Pegasus and Candiru. https://citizenlab.ca/2022/04/catalangate-extensive-mercenary-spyware-operation-against-catalans-using-pegasus-candiru/, 2022. Last visited on 22-05-2022. (Cited on pages 2 and 14.)

[Sta96] Markus Stadler. Publicly Verifiable Secret Sharing. In *EUROCRYPT*, 1996. (Cited on page 93.)

[Ste24] Douglas Stebila. Security analysis of the iMessage PQ3 protocol. Cryptology ePrint Archive, 2024. (Cited on page 135.)

[STV+16] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *IEEE S&P*, 2016. (Cited on pages 59 and 96.)

[Sup22] Signal Support. Twilio Incident: What Signal Users Need to Know. https://support.signal.org/hc/en-us/articles/4850133017242, 2022. Last visited on 03-10-2022. (Cited on page 14.)

[SV21] Sarah Scheffler and Mayank Varia. Protecting Cryptography Against Compelled Self-Incrimination. In *USENIX Security*, 2021. (Cited on pages 6 and 115.)

[TD17] Alin Tomescu and Srinivas Devadas. Catena: Efficient non-equivocation via Bitcoin. In *IEEE S&P*, 2017. (Cited on page 96.)

[Tec19] Weald Technology. go-merkletree. https://github.com/wealdtech/go-merkletree, 2019. (Cited on page 96.)

[TFBT21] Nirvan Tyagi, Ben Fisch, Joseph Bonneau, and Stefano Tessaro. Client-Auditable Verifiable Registries. Cryptology ePrint Archive, 2021. (Cited on page 106.)

[TGL+19] Nirvan Tyagi, Paul Grubbs, Julia Len, Ian Miers, and Thomas Ristenpart. Asymmetric Message Franking: Content Moderation for Metadata-Private End-to-End Encryption. In *CRYPTO*, 2019. (Cited on pages 5 and 115.)

[TLMR22] Nirvan Tyagi, Julia Len, Ian Miers, and Thomas Ristenpart. Orca: Blocklisting in Sender-Anonymous Messaging. In *USENIX Security*, 2022. (Cited on page 135.)

[TPY+19] Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, et al. Protecting accounts from credential stuffing with password breach alerting. In *USENIX Security*, 2019. (Cited on pages 8 and 58.)

[Tub21] Carles Tubio. SKS dump. https://pgp.key-server.io/sks-dump, 2021. Last visited on 7-04-2021. (Cited on page 98.)

[Tur50] Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950. (Cited on page 111.)

[UDB+15] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. Sok: Secure Messaging. In *IEEE S&P*, 2015. (Cited on page 4.)

[UG15] Nik Unger and Ian Goldberg. Deniable key exchanges for secure messaging. In *CCS*, 2015. (Cited on page 111.)

[UG18] Nik Unger and Ian Goldberg. Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging. In *PoPETs*, 2018. (Cited on pages 108 and 111.)

[Vau22] Serge Vaudenay. The Dark Side of SwissCovid. https://lasec.epfl.ch/people/vaudenay/swisscovid.html, 2022. Last visited on 2-10-2024. (Cited on page 6.)

[VGIK20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the Cryptographic Deniability of the Signal Protocol. In *ACNS*, 2020. (Cited on pages 111 and 122.)

[Wan17] Frank Wang. Function Secret Sharing (FSS) Library. https://github.com/frankw2/libfss, 2017. (Cited on page 96.)

[WCGFJ12] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Scalable Anonymous Group Communication in the Anytrust Model. In *EuroSec*, 2012. (Cited on pages 9, 58, and 68.)

[WCWB24] Faxing Wang, Shaanan Cohney, Riad Wahby, and Joseph Bonneau. NOTRY: Deniable messaging with retroactive avowal. *PoPETs*, 2024. (Cited on page 115.)

[Wha24a] WhatsApp. How to check read receipts. https://faq.whatsapp.com/665923838265756/?cms_platform=android, 2024. Last visited on 13-6-2024. (Cited on page 116.)

[Wha24b] WhatsApp. Information for Law Enforcement Authorities. https://faq.whatsapp.com/444002211197967, 2024. Last visited on 23-2-2024. (Cited on page 118.)

[Wik16a] WikiLeaks. DKIM Verification. https://wikileaks.org/DKIM-Verification.html, 2016. Last visited on 07-11-2022. (Cited on page 137.)

[Wik16b] WikiLeaks. Hillary Clinton Email Archive. https://wikileaks.org//clinton-emails/, 2016. Last visited on 07-11-2022. (Cited on pages 126 and 137.)

[WY05] David Woodruff and Sergey Yekhanin. A geometric approach to information-theoretic private information retrieval. In *CCC*, 2005. (Cited on page 105.)

[WYG+17] Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical private queries on public data. In *NSDI*, 2017. (Cited on pages 8, 9, 58, 60, 89, 97, and 105.)

[WZ18] Xingfeng Wang and Liang Zhao. Verifiable Single-Server Private Information Retrieval. In *ICICS*, 2018. (Cited on pages 58, 62, and 106.)

[Yek07] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *STOC*, 2007. (Cited on page 105.)

[YGS23] Tarun Kumar Yadav, Devashish Gosain, and Kent E. Seamons. Cryptographic Deniability: A Multi-perspective Study of User Perceptions and Expectations. In *USENIX Security*, 2023. (Cited on pages 6, 109, 110, 114, 129, 134, 135, 136, and 144.)

[YXB02] Erica Y. Yang, Jie Xu, and Keith H. Bennett. Private Information Retrieval in the Presence of Malicious Failures. In *COMPSAC*, 2002. (Cited on pages 62 and 106.)

[ZHKS16] Ruiyu Zhu, Yan Huang, Jonathan Katz, and Abhi Shelat. The Cut-and-Choose Game and Its Application to Cryptographic Protocols. In *USENIX Security*, 2016. (Cited on page 69.)

[ZMM+20] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. DECO: Liberating Web Data Using Decentralized Oracles for TLS. In *CCS*, 2020. (Cited on page 124.)

[ZS14] Liang Feng Zhang and Reihaneh Safavi-Naini. Verifiable Multi-server Private Information Retrieval. In *ACNS*, 2014. (Cited on pages 9, 58, 62, and 106.)

[ZWH21] Liang Zhao, Xingfeng Wang, and Xinyi Huang. Verifiable single-server private information retrieval from LWE with binary errors. *Inf. Sci.*, 2021. (Cited on pages 58, 62, and 106.)

Simone Colombo

www.simonecolombo.info
si-co

Education

2019–now **Ph.D. in Computer and Communication Sciences**, *Decentralized and Distributed Systems lab, EPFL*, Lausanne, Advisor: Bryan Ford

2015–2018 **Master in Communication Systems with specialization in information security**, *EPFL*, Lausanne

2014–2015 **Erasmus in Communication Systems**, *Technische Universität*, Berlin

2012–2015 **Bachelor in Communication Systems**, *EPFL*, Lausanne

Refereed publications

[4] **Real-world Deniability in Messaging**
D. Collins, S. Colombo, L. Huguenin-Dumittan.
Proceedings on Privacy Enhancing Technologies (PETS), 2025
Real World Crypto Symposium (RWC), 2023

[3] **E-Vote Your Conscience: Perceptions of Coercion and Vote Buying, and the Usability of Fake Credentials in Online Voting**
L.-H. Merino, A. Azhir, H. Zhang, S. Colombo, B. Tellenbach, V. Estrada-Galiñanes, B. Ford
IEEE Symposium on Security and Privacy (Oakland), 2024

[2] **On Active Attack Detection in Messaging with Immediate Decryption**
K. Barooti, D. Collins, S. Colombo, L. Huguenin-Dumittan, S. Vaudenay
CRYPTO, 2023

[1] **Authenticated private information retrieval**
S. Colombo, K. Nikitin, B. Ford, D. Wu, H. Corrigan-Gibbs
USENIX Security Symposium, 2023

Manuscripts

[2] **TRIP: Trust-Limited Coercion-Resistant In-Person Voter Registration**
L.-H. Merino, S. Colombo, R. Reyes, A. Azhir, H. Zhang, J. Allen, B. Tellenbach, V. Estrada-Galiñanes, B. Ford

[1] **Constant-Time Arithmetic for Safer Cryptography**
L. C. Meier, S. Colombo, M. Thiercelin, B. Ford

Teaching assistantship

MATH-106 **Analysis II**, Spring 2020, 2022

COM-402 **Information security and privacy**, Fall 2021

CS-523 **Advanced topics on privacy enhancing technologies**, Spring 2023

COM-500 **Statistical signal and data processing through applications**, Spring 2021

CS-234 **Technologies for democratic society**, Fall 2019, 2020

Supervision

Spring 2024 **Advanced crypto library in Go – Improving Kyber**, M. Suez, R. Goumaz, K. Lauener, Master's project

Fall 2023 **Anamorphic secure messaging**, A. S. Döring, Master's project

Fall 2022 **d-voting security audit**, C. C. Lew, Master's project

Summer 2022 **Cryptographic analysis of TRIP**, R. Reyes, Summer intern

Fall 2021 **3PBCS: A Privacy-Preserving, Personhood-Based Credential System**, K. Apostoli, Master's thesis

Spring 2021 **Constant Time Big Numbers (for Go)**, L. C. Meier, Bachelor's project

Fall 2020 **Streaming Public Key Encryption**, A. Hrusanov, Master's project