Towards Compact Bandwidth and Efficient Privacy-Preserving Computation

A dissertation submitted towards the degree of

Doctor of Natural Sciences

of the Faculty of Mathematics and Computer Science
of Saarland University

by
Sihang Pu

Saarbrücken, 2023

Zusammenfassung

Im Gegensatz zu traditionellen kryptografischen Aufgaben, bei denen Kryptografie verwendet wird, um die Sicherheit und Integrität von Kommunikation oder Speicherung zu gewährleisten und der Gegner typischerweise ein Außenstehender ist, der versucht, die Kommunikation zwischen Sender und Empfänger abzuhören, ist die Kryptografie, die in der datenschutzbewahrenden Berechnung (oder sicheren Berechnung) verwendet wird, darauf ausgelegt, die Privatsphäre der Teilnehmer voreinander zu schützen.

Insbesondere ermöglicht die datenschutzbewahrende Berechnung es mehreren Parteien, gemeinsam eine Funktion zu berechnen, ohne ihre Eingaben zu offenbaren. Sie findet zahlreiche Anwendungen in verschiedenen Bereichen, einschließlich Finanzen, Gesundheitswesen und Datenanalyse. Sie ermöglicht eine Zusammenarbeit und Datenaustausch, ohne die Privatsphäre sensibler Daten zu kompromittieren, was in der heutigen digitalen Ära immer wichtiger wird.

Obwohl datenschutzbewahrende Berechnung aufgrund ihrer starken Sicherheit und zahlreichen potenziellen Anwendungen in jüngster Zeit erhebliche Aufmerksamkeit erregt hat, bleibt ihre Effizienz ihre Achillesferse. Datenschutzbewahrende Protokolle erfordern deutlich höhere Rechenkosten und Kommunikationsbandbreite im Vergleich zu Baseline-Protokollen (d.h. unsicheren Protokollen).

Daher bleibt es eine spannende Aufgabe, Möglichkeiten zu finden, um den Overhead zu minimieren (sei es in Bezug auf Rechen- oder Kommunikationsleistung, asymptotisch oder konkret), während die Sicherheit auf eine angemessene Weise gewährleistet bleibt.

Diese Arbeit konzentriert sich auf die Verbesserung der Effizienz und Reduzierung der Kosten für Kommunikation und Berechnung für gängige datenschutzbewahrende Primitiven, einschließlich private Schnittmenge, vergesslicher Transfer und Stealth-Signaturen. Unser Hauptaugenmerk liegt auf der Optimierung der Leistung dieser Primitiven.

Thesis advisor: Dr. Nico Döttling Sihang Pu

Towards Compact Bandwidth and Efficient Privacy-Preserving Computation

Abstract

In traditional cryptographic applications, cryptographic mechanisms are employed to ensure the security and integrity of communication or storage. In these scenarios, the primary threat is usually an external adversary trying to intercept or tamper with the communication between two parties. On the other hand, in the context of privacy-preserving computation or secure computation, the cryptographic techniques are developed with a different goal in mind: to protect the privacy of the participants involved in a computation from each other.

Specifically, privacy-preserving computation allows multiple parties to jointly compute a function without revealing their inputs and it has numerous applications in various fields, including finance, healthcare, and data analysis. It allows for collaboration and data sharing without compromising the privacy of sensitive data, which is becoming increasingly important in today's digital age.

While privacy-preserving computation has gained significant attention in recent times due to its strong security and numerous potential applications, its efficiency remains its Achilles' heel. Privacy-preserving protocols require significantly higher computational overhead and bandwidth when compared to baseline (i.e., insecure) protocols.

Therefore, finding ways to minimize the overhead, whether it be in terms of computation or communication, asymptotically or concretely, while maintaining security in a reasonable manner remains an exciting problem to work on.

This thesis is centred around enhancing efficiency and reducing the costs of communication and computation for commonly used privacy-preserving primitives, including private set intersection, oblivious transfer, and stealth signatures. Our primary focus is on optimizing the performance of these primitives.

Contents

To my parents who support me all the time.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Nico Döttling, for his steadfast mentorship and advice, which guided me in my research and cultivated my professional growth. His insightful feedback, patience, and intellectual rigor have been instrumental in shaping this work and my academic career.

I also wish to express my sincere appreciation to Antoine Joux, Markus Bläser, and my exceptional co-referee, Geoffroy Couteau. Their willingness to review my work despite their own demanding schedules is something I deeply appreciate.

Special thanks go to my collaborators, particularly Pedro Branco, and Sri Aravinda Krishman, with whom I had the privilege of collaborating on several research projects. Their unique perspectives, insightful contributions, and dedication to high-quality research have significantly enriched my work and personal growth as a researcher. I also wish to acknowledge my colleagues Jesko Dujmovic, Chuan-wei Lin, and Nabil Alkeilani Alkadri, who created an inspiring work environment and provided much-needed support and motivation throughout this journey. Their camaraderie, intellectual discussions, and shared advice were invaluable in the completion of this thesis.

Lastly, I would like to express my heartfelt gratitude to my family and friends, whose unwavering belief in my abilities, constant support, and endless encouragement have been the cornerstone of my resilience during the entirety of this scholarly pursuit. Thank you for being my source of strength and positivity.

0 Introduction

Privacy-preserving computation, or secure computation, is an extension of traditional cryptography to protect users' sensitive information and metadata during computation.

In contrast to traditional cryptographic tasks, where cryptography is used to ensure the security and integrity of communication or storage, and the adversary is typically an outsider attempting to eavesdrop on the communication between sender and receiver, the cryptography used in the privacy-preserving computation is designed to also safeguard the privacy of the participants from one another.

Specifically, privacy-preserving computation allows multiple parties to jointly compute a function without revealing their inputs and it has numerous applications in various fields, including finance, healthcare, and data analysis. It allows for collaboration and data sharing without compromising the privacy of sensitive data, which is becoming increasingly important in today's digital age. For a concrete example, imagine a medical center that holds a potentially vast database of disease-associated genetic variants. A patient wants to undergo a DNA screening to check for any issues. However, due to privacy concerns, the patient is hesitant to provide their DNA in plain form. This is where privacy-preserving computation comes into play. Both parties (the medical center and the patient) can cooperatively execute a secure protocol, allowing the patient to learn if they have any genetic markers associated with diseases without revealing additional confidential information about their DNA.

Motivation

Despite the burgeoning interest and potential applications associated with privacy-preserving computation, owing to its robust security, the *efficiency* of such protocols continues to be a challenge. Privacy-preserving protocols require significantly higher computational overhead and bandwidth when compared to baseline (i.e., insecure) protocols.

As an example, consider the secure computation functionality of the set intersection (i.e., private set intersection), which is one of the most popular applications of privacy-preserving computation.

In this scenario, Alice possesses a large dataset A, while Bob holds a much smaller dataset B such that $|B| \ll |A|$. Alice wants to determine if there is any intersection between their datasets without learning anything about Bob's dataset or revealing any information about her own.

The baseline protocol for set intersection involves Bob sending his dataset in plain text, which Alice then compares to her own dataset to identify any common elements. Note that this approach entails $O(|B|)$ communication bandwidth, which is inherent to this method.

In contrast, secure computation protocols usually entail $O(|A| + |B|)$ communication bandwidth. While it ensures that the privacy of both parties is protected, and no information is revealed to either party beyond what is necessary for the set intersection computation, this approach incurs a prohibitively higher communication cost if $|A|$ is super large. For instance, a DNS server holding a 100 GB set with each entry 4 Bytes is a normal case but 100 GB communication bandwidth is highly undesired!

It is important to note that the efficiency gap between traditional and secure computation protocols is not just limited to asymptotic measures but is also evident in concrete scenarios. For example, consider a basic functionality (i.e., oblivious transfer) where a receiver holds a bit $b \in \{0, 1\}$ and a sender holds two bits x_0 and x_1. The goal is for the receiver to learn x_b while remaining ignorant of x_{1-b}, and the sender must not know anything about the value of b. Baseline insecure protocols only require a bandwidth of *two bits*, while secure protocols often require two ciphertexts, each with a size of approximately *512* bits if implemented using the most efficient 256-bit elliptic curves.

Therefore, finding ways to minimize the overhead (whether it be in terms of computation or communication, asymptotically or concretely) while maintaining security in a reasonable manner remains an exciting problem to work on.

This thesis is centred around enhancing efficiency and reducing the costs of communication and computation for commonly used privacy-preserving primitives. We tackled several problems in privacy-preserving computation, the first of which involves multiparty threshold private set intersection (PSI). For threshold PSI, the parties involved in the protocol learn the output if the size of the intersection between the input sets of the parties is very large, say larger than $n - t$, where n is the size of the input sets and t is some *threshold* such that $t \ll n$; Otherwise, they learn nothing about the intersection. This is in contrast with standard PSI where the parties always get the intersection, no matter its size. The main reason for considering this problem (apart from its numerous applications like ride-sharing, contact discovery etc.) is that the amount of communication needed is much smaller than for standard PSI: In particular, there are threshold PSI protocols whose communication complexity depends only on the threshold t and not on the size of the input sets as for standard PSI [GS19a] in two-party setting. However, non-trivial threshold PSI protocols in the multiparty setting are still an open question to solve. Because if someone naively extends [GS19a] to a multiparty setting then the bandwidth *for each participant* will depend on the number of parties which is prohibitively expensive.

The second problem we investigated is unbalanced PSI, as briefly mentioned at the outset. In this scenario, two parties aim to compute the intersection of their respective sets, but one set significantly outnumbers the other in terms of its size. Solving this problem led us to initiate the study of laconic PSI: Laconic PSI allows a receiver to send a short digest of its *large* data set, which in turn can be used by potentially many different senders to compute a PSI second-round message. We require that the total communication complexity as well as the sender's running time be independent of the receiver's input size. Though a non-black-box approach is known via general purpose laconic function evaluation [QWW18], we are interested in providing a *black-box* solution for efficiency concern. The black-box solution refers to the construction without using any explicit circuit-level description of cryptographic primitives. Particularly, we consider constructions which compute cryptographic primitives inside garbled circuits or express statements in terms of NP-complete languages as non-black-box approaches, which are notably demanding, resulting in either substantial computational costs or significant communication overheads.

The third problem we addressed concerns the rate of oblivious transfer (OT). In most applications, one OT is not enough and it is required to perform many OT operations in parallel. We let n denote the number of parallel executions. Various techniques have been developed to address this task of *batch-OT* [IKNP03, BCG$^+$19b, BCG$^+$19a]. For the most part, they involve a preprocessing "offline" phase where the parties generate random OT correlations. Given such correlations, executing the OT protocol in the so-called "online phase" is computationally very simple. This approach is very useful for purposes of computational efficiency since the offline phase can be carried out even before the actual inputs of the computation are known. However, in terms of communication complexity, there is an inherent cost, even just in the online phase, of n receiver bits and $2n$ sender bits. In contrast, the insecure implementation only requires n bits to be sent from each party in a two-message protocol: the receiver sends its input, and the sender returns all of the appropriate x_b values. As always in cryptography, we wish to understand what is the "cost of privacy", namely how closely can we approach the information-theoretic minimum without losing privacy.

The final problem we addressed pertains to *stealth signatures*, which we introduced in the following manner. In this scenario, the receiver generates a master key pair and disseminates the master public key. Any sender can then *locally re-randomize* this master public key into a one-time public key. For any external observer, this one-time public key is *unlinkable* to the master public key. However, when the receiver has access to the master secret key, it can *link* this one-time public key to its master public key, and also generate the corresponding one-time secret key locally, on-the-fly. Utilizing this one-time secret key, the receiver can sign messages without revealing its metadata, meaning that an external observer will not be able to ascertain which public key matches the signature. It's important to note in this mechanism, the receiver only needs to broadcast its master public key, and *does not* need to distribute a distinct unlinkable one-time public key for each potential sender. Given that the number of senders could potentially reach hundreds or thousands, this feature offers significant benefits. This mechanism is extensively employed in privacy-preserving cryptocurrencies such as Monero, and also has applications in passwordless authentication as defined in the Fast IDentity Online (FIDO) standard. However, the current known protocols for this mechanism are either insecure in some reasonable adversarial models or inefficient in practical implementation.

At the end of the introduction, this thesis is organized as follows:

Contributions and roadmaps

- In Chapter 1, we provide essential preliminaries about basic cryptography primitives, security definitions or frameworks, well-established assumptions, lattices, polynomials, and some statistical tools.
- In Chapter 2, we present a protocol of multiparty threshold private set intersection, which improves communication bandwidth *for each party* from $\tilde{O}(Nt^2)$ to $\tilde{O}(t^2)$ where N is the number of parties and t the threshold while retaining the same computational overhead and security level.
- In Chapter 3, we introduce a new primitive, laconic private set intersection, which solves unbalanced PSI in a non-interactive way while making communication bandwidth as succinct as possible. Specifically, after the server publishes a short digest of constant size, any client can non-interactively send its message of size independent of the server's dataset.
- In Chapter 4, we present a two-message oblivious transfer protocol which has asymptotically minimum communicational bandwidth, namely, to transfer n bits information, it only requires $n(1+o(1))$ bits bandwidth for each user while retaining computational efficiency. We also show how to efficiently emulate $\mathbb{Z}_2$ inside a prime-order group $\mathbb{Z}_p$ in a function-private manner.
- In Chapter 5, we present a post-quantum privacy-preserving signature called stealth signature that saves 70% bandwidth compared to the state of the art while achieving the strongest security. Additionally, we present a fuzzy variant which protects users' metadata and improves the server's computational work from $O(N)$ to $O(\sqrt{N})$ where N is the number of users.
- In Chapter 6, we summarise the thesis in a coherent and concise way.

1 Preliminary

In this preliminary chapter, we lay the groundwork for our exploration by introducing essential concepts, setting the stage for a comprehensive understanding of the subject matter as we delve into the subsequent chapters.

We denote by $\lambda \in \mathbb{N}$ the security parameter, by $\mathsf{poly}(\lambda)$ any function that is bounded by a polynomial in λ, and by $\mathsf{negl}(\lambda)$ any function that is negligible in the security parameter. We abbreviate the computational indistinguishability of two distributions by $\approx_c$. The set of N elements is always written as $[N]$. We also denote as $\mathcal{D}^{\mathcal{O}}$ a distinguisher $\mathcal{D}$ access to an oracle $\mathcal{O}$ via classical queries and $\mathcal{A}^{|\mathcal{O}\rangle}$ via quantum queries. If S is a finite set, then $x \leftarrow\!\$\ S$ denotes an element x sampled from S according to a uniform distribution and $|S|$ denotes the cardinality of S. For two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$ over a finite field $\mathbb{F}$, we denote by $\mathbf{u} \odot \mathbf{v}$ their component-wise multiplication. We denote by $\mathsf{Supp}(\mathbf{u})$ the support of $\mathbf{u}$, that is, the set of indices where $\mathbf{u}$ is different from 0.[1] For $S \subseteq [n]$, $\mathbf{u}_S$ denotes the vector $(\mathbf{u}_i)_{i \in S}$. Finally, $\mathbf{u}^T$ denotes the transpose of $\mathbf{u}$ and $\mathsf{hw}(\mathbf{u})$ denotes the hamming weight of $\mathbf{u}$ (that is, the number of coordinates of $\mathbf{u}$ different from 0).

Definition 1.0.1 (Statistical Distance). The statistical distance between two probability distributions A and B is

$$\mathsf{SD}(A, B) = \frac{1}{2} \sum_{v} \big| \Pr[A = v] - \Pr[B = v] \big|.$$

Recall min-entropy of a random variable A is

$$H_\infty(A) := -\log(\max_{a} \Pr[A = a]),$$

then we have the following lemma.

[1] If there is only one index different from zero, $\mathsf{Supp}(\mathbf{u})$ denotes this index.

Lemma 1.0.1 (Leftover Hash Lemma[ILL89]). *Assume a family of functions* $\{\mathsf{H}_x : \{0,1\}^n \mapsto \{0,1\}^m\}_{x\in\mathcal{X}}$ *is universal:* $\forall a \neq b \in \{0,1\}^n$, $\Pr_{x\in\mathcal{X}}[\mathsf{H}_x(a) = \mathsf{H}_x(b)] = 2^{-m}$. *Then, for any random variable W,*

$$\mathsf{SD}((\mathsf{H}_X(W), X), (U_m, X)) \leq \varepsilon,$$

whenever $m \leq k - 2\log(\frac{1}{\varepsilon}) + 2$ *and* $k = H_\infty(W)$.

Lemma 1.0.2 (Rank of the Circulant Matrix[Ing56]). *The rank of a circulant matrix* C *of order* m *is* $m - d$, *where* d *is the degree of the greatest common divisors of* $X^m - 1$ *and the associated polynomial of* C.

Here, we present the cryptographic primitives and definitions which are meaningful to this thesis, as well as their security properties.

1.1 Basic Primitives

1.1.1 Digital Signatures

A digital signature scheme DS, formally, has a key generation algorithm $\mathsf{KGen}(\lambda)$ that takes the security parameter λ and outputs the verification/signing key pair $(\mathsf{vk}, \mathsf{sk})$, a signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ inputs a signing key and a message $m \in \{0,1\}^*$ and outputs a signature σ, and a verification algorithm $\mathsf{Vf}(\mathsf{vk}, m, \sigma)$ outputs 1 if σ is a valid signature on m under the verification key vk, and outputs 0 otherwise. We require unforgeability, which guarantees that a PPT adversary cannot forge a fresh signature on a fresh message of its choice under a given verification key while having access to a signing oracle (that returns valid signatures on the queried messages). Formally the notion can be captured in an experiment denoted by EUF-CMA. Strong unforgeability refers to the case where the adversary is required to forge a fresh signature on not necessarily a fresh message. Formally the notion can be captured in an experiment denoted by sEUF-CMA.

1.1.2 Key Encapsulation Mechanism

A key encapsulation mechanism KEM, formally, has a key generation algorithm $\mathsf{KGen}(\lambda)$ that takes the security parameter λ and outputs a encaps key ek and a decaps key dk. An encapsulation algorithm $\mathsf{Encaps}(\mathsf{ek})$ inputs an encaps key and outputs a ciphertext C and agreed key K. Finally, we have a decapsulation algorithm $\mathsf{Decaps}(\mathsf{dk})$ inputs a decaps key and a ciphertext and outputs an agreed key K. Apart from IND-CCA security, we additionally require its anonymous property which can be formally captured in Definition 1.1.5 denoted by ANO-CCA and it means the adversary cannot link any ciphertext C to its encaps key ek even being able to access a decaps oracle. Concretely, we use Kyber [SAB+20] with the modification shown in Figure 6 of [GMP22].

1.1.3 UC Framework

In this thesis, we use the UC framework by Canetti [Can01] to analyze the security of our protocols.[2] Throughout this thesis, we usually consider semi-honest adversaries, unless stated otherwise. Let $\mathcal{F}$ be a functionality, π a protocol that implements $\mathcal{F}$ and $\mathcal{E}$ be an environment, an entity that oversees the execution of the protocol in both the real and the ideal worlds. Let $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}}$ be a random variable that represents the output of $\mathcal{E}$ after the execution of $\mathcal{F}$ with adversary Sim. Similarly, let $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{E}}$ be a random variable that represents the output of $\mathcal{E}$ after the execution of π with adversary $\mathcal{A}$.

Definition 1.1.1. A protocol π *implements* $\mathcal{F}$ if for every PPT adversary $\mathcal{A}$ there is a PPT simulator Sim such that for all PPT environments $\mathcal{E}$, the distributions $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}}$ and $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{E}}$ are computationally indistinguishable.

1.1.4 Strong Extractors

Extractors allow the extraction of randomness from sources with a certain min-entropy.

Definition 1.1.2 (Strong Extractor)**.** A (k,ε)*-strong extractor* $\mathsf{Ext} : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ is a deterministic algorithm with domain $\mathcal{X}$, seed space $\mathcal{S}$ and range $\mathcal{Y}$ with the following property: For every distribution X with support $\mathcal{X}$ and min-entropy at least k,

$$(s, \mathsf{Ext}(s,x)) \approx_{\varepsilon} (s, y)$$

where $x \leftarrow\$ X$ and $y \leftarrow\$ \mathcal{Y}$.

1.1.5 Public-Key Encryption

We recall the classical definition of public-key encryption (PKE).

Definition 1.1.3 (Public-Key Encryption)**.** A *Public-Key Encryption* (PKE) scheme is defined by the following algorithms:

- $\mathsf{KeyGen}(1^{\lambda})$ takes as input a security parameter. It outputs a public key pk and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, m)$ takes as input a public key pk and a message $m \in \{0,1\}^*$. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ takes as input a secret keys sk and a ciphertext ct. It outputs a message m or bot $\perp$.

We require the usual correctness and IND-CPA properties for a PKE.

[2]We refer the reader to [Can01] for a detailed explanation of the framework.

```
$\text{ANO-CCA}^{\mathcal{A}}_{\mathsf{KEM}}(\lambda)$
    $(\mathsf{ek}_0, \mathsf{dk}_0) \leftarrow \mathsf{KEM.Gen}(\lambda)$
    $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{KEM.Gen}(\lambda)$
    $b \leftarrow\!\!\$\ \{0,1\}$
    $(C^*, K^*) \leftarrow \mathsf{KEM.Encaps}(\mathsf{ek}_b)$
    $b' \leftarrow \mathcal{A}^{\mathsf{Decaps}\mathcal{O}(\cdot,\cdot)}(\mathsf{ek}_0, \mathsf{ek}_1, C^*, K^*)$
    $b_0 := (b = b')$
    return $b_0$

$\mathsf{Decaps}\mathcal{O}(b', C')$
    $K' := \mathsf{KEM.Decaps}(\mathsf{dk}_{b'}, C')$
    return $K'$
```

Figure 1.1: Experiment for $\text{ANO-CCA}^{\mathcal{A}}_{\mathsf{KEM}}(\lambda)$

- **Correctness:** We say that a PKE is *correct* if

$$\Pr\left[m \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) : \ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\ \right] = 1.$$

- **IND-CPA security:** For any PPT adversary $\mathcal{A}$, we require that

$$\Pr\left[b \leftarrow \mathcal{A}(\mathsf{ct}, \mathsf{st}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda);\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pk}) \\ b \leftarrow\!\!\$\ \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 1.1.4 (Binomial Distribution[SAB+20]). We define the binomial distribution B_η as follows:

$$(a_1, \ldots, a_\eta, b_1, \ldots, b_\eta) \leftarrow\!\!\$\ \{0,1\}^{2\eta},$$

and then output $\sum_i^\eta a_i - b_i$. If we write some polynomial $f \leftarrow\!\!\$\ B_\eta$, then each coefficient of f is sampled from B_η.

Definition 1.1.5 (Anonymous KEM[GMP22]). A KEM is said to be *anonymous under chosen-ciphertext attacks* if there exists a negligible function $\mathsf{negl}(\lambda)$ for all $\lambda \in \mathbb{N}$, and for all adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\text{ANO-CCA}^{\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where ANO-CCA is defined in Figure 1.1. Similarly, we also define IK-CPA experiment for PKE in Figure 1.2, which just removes access to the decryption oracle[BBDP01].

Definition 1.1.6 (Uniformly-Ambiguous Encryption[BLMG21]). Let $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme for the message space $\{0,1\}^n$. For any $\lambda \in \mathbb{N}$, uniformly sampled message $\mathbf{m} \leftarrow\!\!\$\ \{0,1\}^n$, we say PKE is UNI-AMB-secure if

$$\mathsf{Adv}^{\text{UNI-AMB}}_{\lambda}(\mathcal{A}) := \Big| \Pr[\text{UNI-AMB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda) \Rightarrow 0] - \Pr[\text{UNI-AMB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda) \Rightarrow 1] \Big| \leq \mathsf{negl}(\lambda),$$

```
$\mathrm{UNI\text{-}AMB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{PKE.Gen}(\lambda)$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{PKE.Gen}(\lambda)$
$b \leftarrow\!\!\$ \{0,1\}$
$\mathbf{m} \leftarrow\!\!\$ \{0,1\}^n$
$\mathbf{c}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_b, \mathbf{m})$
$b' \leftarrow \mathcal{A}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{sk}_1, \mathbf{c}^*)$
**return** $b \overset{?}{=} b'$
```

```
$\mathrm{IK\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{PKE.Gen}(\lambda)$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{PKE.Gen}(\lambda)$
$(\mathsf{st}_{\mathcal{A}}, m) \leftarrow \mathcal{A}_1(\mathsf{pk}_0, \mathsf{pk}_1)$
$b \leftarrow\!\!\$ \{0,1\}$
$c^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_b, m)$
$b' \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \mathsf{pk}_0, \mathsf{pk}_1, c^*)$
**return** $b \overset{?}{=} b'$
```

Figure 1.2: Experiment for $\mathrm{UNI\text{-}AMB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)$ and $\mathrm{IK\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)$

where the experiment $\mathrm{UNI\text{-}AMB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)$ is defined in Figure 1.2.

1.2 Polynomials

We first introduce minimal polynomials of a sequence and of a matrix. Then we present how they can be used to solve linear algebra-related problems.

Minimal Polynomial of a Matrix

The minimal polynomial of a sequence $\mathfrak{a}$ is the least degree polynomial m such that $\langle m \rangle = Ann(\mathfrak{a})$ where $Ann(\mathfrak{a})$ is the annihilator ideal of $\mathfrak{a}$ (that is, the ideal such that every element f of $Ann(\mathfrak{a})$ satisfies $f \cdot \mathfrak{a} = 0$).

Lemma 1.2.1 (Lemma 3 in [KMWF07]). *Let $\mathbf{A} \in \mathbb{F}^{n\times n}$ and let $m_{\mathbf{A}}$ be the minimal polynomial of matrix $\mathbf{A}$. For $\mathbf{u}, \mathbf{v} \leftarrow\!\!\$ \mathbb{F}^n$, we have $m_{\mathbf{A}} = m_{\mathfrak{a}'}$ with probability at least $1 - 2\deg(m_{\mathbf{A}})/|\mathbb{F}|$, where $\mathfrak{a}' = (\mathbf{u}^\top \mathbf{A}^i \mathbf{v})_{i\in\mathbb{N}}$. Moreover, $m_{\mathfrak{a}'}$ can be calculated using a Boolean circuit of size $\mathcal{O}(nk \log n \log k \log\log k)$ where $k = \log|\mathbb{F}|$*

Compute the Rank of a Matrix and Solve a Linear System

Lemma 1.2.2 ([KDS91]). *Let $\mathbf{A} \in \mathbb{F}^{n\times n}$ of (unknown) rank r. Let $\mathbf{U}$ and $\mathbf{Z}$ be randomly chosen unit upper triangular and lower triangular Toeplitz matrices in $\mathbb{F}^{n\times n}$, and let $\mathbf{B} = \mathbf{UAZ}$. Let us denote the $i \times i$ leading principal of $\mathbf{B}$ by $\mathbf{B}_i$. The probability that $\det(\mathbf{B}_i) \neq 0$ for all $1 \leq i \leq r$ is greater than $1 - n^2/|\mathbb{F}|$.*

Lemma 1.2.3 ([KDS91]). *Let $\mathbf{B} \in \mathbb{F}^{n\times n}$ with leading invertible principals up to $\mathbf{B}_r$ where r is the (unknown) rank of $\mathbf{B}$. Let $\mathbf{X}$ be a randomly chosen diagonal matrix in $\mathbb{F}^{n\times n}$. Then, $r = \deg(m_{\mathbf{XB}}) - 1$ with probability greater than $1 - n^2/|\mathbb{F}|$.*

1.2.1 Polynomials and Interpolation

We present a series of results that will be useful to analyze the correctness and security of the protocols presented in this thesis.

The following lemma shows how we can mask a polynomial of degree less than t using a uniformly random polynomial.

Lemma 1.2.4 ([KS05]). *Let $\mathbb{F}_p$ be a prime order field, $P(x), Q(x)$ be two polynomials over $\mathbb{F}_p$ such that $\deg P = \deg Q = d \leq t$ and $\gcd(P, Q) = 1$. Let $R_1, R_2 \leftarrow\$ \mathbb{F}_p$ such that $\deg R_1 = \deg R_2 = t$. Then $U(x) = P(x)R_1(x) + Q(x)R_2(x)$ is a uniformly random polynomial with $\deg U \leq 2t$.*

Note that this result also applies to multiple polynomials as long as they don't share a common factor (referring to Theorem 2 and Theorem 3 of [KS05] for more details).

We say that f is a rational function if $f(x) = \frac{P(x)}{Q(x)}$ for two polynomials P and Q.

The next two lemmata show that we can recover a rational function via interpolation and that this function is unique.

Lemma 1.2.5 ([MTZ03]). *Let $f(x) = P(x)/Q(x)$ be rational function where $\deg P(x) = m$ and $\deg Q(x) = n$. Then $f(x)$ can be uniquely recovered (up to constants) via interpolation from $m + n + 1$ points. In particular, if $P(x)$ and $Q(x)$ are monic, $f(x)$ can be uniquely recovered from $m + n$ points.*

Lemma 1.2.6 ([MTZ03]). *Choose V to be a support set[3] of cardinality $m_1 + m_2 + 1$. Then, there is a unique rational function $f(x) = P(x)/Q(x)$ that can be interpolated from V, and $P(x)$ has degree at most m_1 and $Q(x)$ has degree at most m_2.*

1.3 Lattices

We now review some basic notions of lattices and Gaussian distributions.

Let $\mathbf{B} \in \mathbb{R}^{k \times n}$ be a matrix. We denote the lattice generated by $\mathbf{B}$ by $\Lambda = \Lambda(\mathbf{B}) = \{\mathbf{x}\mathbf{B} : \mathbf{x} \in \mathbb{Z}^k\}$.[4] The dual lattice Λ^* of a lattice Λ is defined by $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \forall y \in \Lambda, \mathbf{x} \cdot \mathbf{y} \in \mathbb{Z}\}$. It holds that $(\Lambda^*)^* = \Lambda$. The orthogonal lattice $\Lambda_q^{\perp}$ is defined by $\{\mathbf{y} \in \mathbb{Z}_q^n : \mathbf{A}\mathbf{y}^T = 0 \mod q\}$.

Definition 1.3.1 (Cyclotomic Polynomial). We denote by R the ring $\mathbb{Z}[X]/(X^n + 1)$ and by R_q the ring $\mathbb{Z}_q[X]/(X^m + 1)$, where $m = 2^{m'-1}$ such that $X^m + 1$ is the $2m'$-th cyclotomic polynomial $\Phi_{2m'}(X)$. Moreover, we have

$$\prod_{d|m} \Phi_d(X) = X^m - 1.$$

Let $\rho_s(\mathbf{x})$ be the probability distribution of the Gaussian distribution over $\mathbb{R}^n$ with parameter s and centred in 0. We define the discrete Gaussian distribution $D_{S,s}$ over S and with parameter s by the probability distribution $\rho_s(\mathbf{x})/\rho(S)$ for all $\mathbf{x} \in S$ (where $\rho_s(S) = \sum_{\mathbf{x} \in S} \rho_s(\mathbf{x})$).

[3] A support set is a set of pairs (x, y).

[4] The matrix $\mathbf{B}$ is called a basis of $\Lambda(\mathbf{B})$.

For $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ of a lattice Λ is the least real $\sigma > 0$ such that $\rho_{1/\sigma}(\Lambda^* \setminus \{0\}) \leq \varepsilon$ [MR04].

Lemma 1.3.1 ([Ban93]). *For all $\alpha \in \mathbb{R}$, $\|\mathbf{x}\| \leq \alpha\sqrt{n}$ for $\mathbf{x} \leftarrow\$ D^n_{\mathbb{Z},\alpha}$, except with negligible probability in n.*

We will make use of the following convolution property of discrete Gaussians.

Lemma 1.3.2 ([GMPW20], Corollary 4.8). *Let $\Lambda_1, \Lambda_2 \subseteq \mathbb{R}^n$ be lattices, let $\sigma_1, \sigma_2 > 0$ be such that $1/\sqrt{1/\sigma_1^2 + 1/\sigma_2^2} > \eta_\varepsilon(\Lambda_1 \cap \Lambda_2)$ for some $\varepsilon = \mathsf{negl}(\lambda)$. Then it holds for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ that $D_{\Lambda_1+\mathbf{a},\sigma_1} + D_{\Lambda_2+\mathbf{b},\sigma_2}$ is statistically close to $D_{\Lambda_1+\Lambda_2+\mathbf{a}+\mathbf{b},\sqrt{\sigma_1^2+\sigma_2^2}}$.*

We just need the following simple corollary of Lemma 1.3.2, which can be obtained by setting $\Lambda_1 = \Lambda_2 = \mathbb{Z}$.

Corollary 1.3.3. *Let $\sigma_1, \sigma_2, \sigma_3 = \sqrt{\sigma_1^2 + \sigma_2^2}$ be such that $\sigma_1\sigma_2/\sigma_3 > \eta_\varepsilon(\mathbb{Z})$ for a negligible ε and let $a, b \in \mathbb{Z}$. Then $D_{\mathbb{Z}+a,\sigma_1} + D_{\mathbb{Z}+b,\sigma_2}$ and $D_{\mathbb{Z}+a+b,\sigma_3}$ are statistically close.*

Gadget matrix. For given parameters $n, q \in \mathbb{Z}$, let $\mathbf{g}$ be the vector $\left(1, 2, 2^2, \ldots, 2^{\lceil \log q \rceil - 1}\right)$ and $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n$ where $\mathbf{I}_n$ is the identity matrix of size n. The matrix $\mathbf{G}$ is usually called the gadget matrix [MP12].

Moreover, let $\bar{\mathbf{G}}_n = \sum_i \mathbf{G}_i \in \mathbb{Z}^{n \times \lceil \log q \rceil}$ where $\mathbf{G}_i$ is the matrix which is zero everywhere but its i-th row is $\mathbf{g}$.

The function $\mathbf{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}^m$, where $m = \lceil \log q \rceil$, receives a value $v \in \mathbb{Z}_q$ and outputs its binary decomposition. Note that $\mathbf{g} \cdot \mathbf{g}^{-1}(v) = v \mod q$. Following [BdMW16], we define $\mathbf{g}^{-1}_{\mathsf{rnd}}$ to be the function that, on input $v \in \mathbb{Z}_q$, outputs $\mathbf{x} \leftarrow\$ D_{\Lambda_q^\perp(\mathbf{g})+\mathbf{g}^{-1}(v),r}$, where $r = \tilde{O}(1)$. It holds that $\mathbf{g} \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}(v) = v \mod q$.

1.4 Hardness Assumptions

We start by introducing some notation. Let $\mathsf{Primes}(\kappa)$ denote the set of prime numbers of bit-length κ. Let

$$\mathsf{RSA}(\lambda) = \{N : N = PQ \text{ and } P, Q \in \mathsf{Primes}(\lambda/2) \text{ and } \gcd(P-1, Q-1) = 2\}$$

and

$$\mathsf{RSA}_e(\lambda) = \{N : e | \varphi(N)\}$$

for any $e \leq 2^\lambda$.

1.4.1 Phi-Hiding

Definition 1.4.1 (Phi-Hiding)**.** The *phi-hiding* assumption, denoted as φ-hiding, states that for all $\varepsilon > 0$ and $3 < e < 2^{\lambda/4-\varepsilon}$ and all PPT adversaries $\mathcal{A}$, we have that

$$|\Pr[1 \leftarrow \mathcal{A}(N,e) : N \leftarrow\!\!\$\ \mathsf{RSA}(\lambda)] - \Pr[1 \leftarrow \mathcal{A}(N,e) : N \leftarrow\!\!\$\ \mathsf{RSA}_e(\lambda)]| \leq \mathsf{negl}(\lambda).$$

Let $N = PQ$ where $\gcd(P-1, Q-1) = 2$. Consider the multiplicative group $\mathbb{Z}^*_{N^{\xi+1}}$ where ξ is a fixed non-negative integer. Recall that $\mathbb{Z}^*_{N^{\xi+1}}$ can be written as the product of two subgroups $\mathbb{H}_N \times \mathbb{NR}_N$ where $\mathbb{H}_N = \{(1+N)^i : i \in [N^\xi]\}$ and $\mathbb{NR}_N = \{x^{N^\xi} : x \in \mathbb{Z}^*_{N^{\xi+1}}\}$ (the subgroup of N^ξ-residues) which has order $\varphi(N)$. Given $(1+N)^m \bmod N^{\xi+1}$, there is a polynomial-time algorithm that allows to recover m [DJ01].

Furthermore, note that $\mathbb{NR}_N$ can be decomposed into the product of two subgroups cyclic $\mathbb{Z}^*_P$ (of order $P-1$) and $\mathbb{Z}^*_Q$ (of order $Q-1$). Since $\gcd(P-1, Q-1) = 2$, then there is a cyclic subgroup $\mathbb{T}_N$ of $\mathbb{Z}^*_P \times \mathbb{Z}^*_Q$ of order $\varphi(N)/2$. Also, consider the product $\mathsf{J}_N = \mathbb{H}_N \times \mathbb{T}_N$. It is easy to show that the subset membership problem for $(\mathbb{J}_N, \mathbb{T}_N)$ is still hard if the DCR assumption holds.

The following lemma is straightforwardly adapted from [GVW20].

Lemma 1.4.1 ([GVW20])**.** *Assume that the φ-hiding assumption holds. Let* Ext *be a* $(\kappa-1, \mathsf{negl}(\lambda))$*-strong extractor. For every admissible stateful PPT adversary $\mathcal{A}$ and for all λ, κ such that $\lambda \geq 5\kappa$, we have that*

$$\left| \Pr\left[b \leftarrow \mathcal{A}(y_b) : \begin{array}{c} N \leftarrow\!\!\$\ \mathsf{RSA}(\lambda);\ s \leftarrow\!\!\$\ \{0,1\}^\lambda \\ e \leftarrow\!\!\$\ \mathsf{Primes}(\kappa);\ g \leftarrow \mathbb{T}_N \\ G \leftarrow \mathcal{A}(N,s,e,g);\ b \leftarrow\!\!\$\ \{0,1\} \\ y_0 \leftarrow \mathsf{Ext}(s, g^{Ge^{-1}} \bmod N^{\xi+1});\ y_1 \leftarrow\!\!\$\ \mathcal{Y} \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

where an admissible adversary is one that outputs G such that e does not divide G.

1.4.2 Decisional Composite Residuosity

In this thesis, we also make use of the Decisional Composite Residuosity (DCR) assumption which we define in the following. We present the DCR assumption as a subgroup indistinguishability assumption [BG10].

Definition 1.4.2 (Decisional Composite Residuosity)**.** Let $N = \mathsf{RSA}(\lambda)$ and let $\xi \geq 0$ be a fixed integer. The *decisional composite residuosity* (DCR) assumption states that for all PPT adversaries $\mathcal{A}$,

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(N,x) : x \leftarrow\!\!\$\ \mathbb{Z}^*_{N^{\xi+1}}\right] - \Pr[1 \leftarrow \mathcal{A}(N,x) : x \leftarrow\!\!\$\ \mathbb{NR}_N]\right| \leq \mathsf{negl}(\lambda).$$

Lemma 1.4.2 ([CS02])**.** *$N = \mathsf{RSA}(\lambda)$ and let $\xi \geq 0$ be a fixed integer. Assume that the DCR assumption holds. Then for all PPT adversaries $\mathcal{A}$,*

$$|\Pr[1 \leftarrow \mathcal{A}(N,x) : x \leftarrow\!\!\$\ \mathbb{J}_N] - \Pr[1 \leftarrow \mathcal{A}(N,x) : x \leftarrow\!\!\$\ \mathbb{T}_N]| \leq \mathsf{negl}(\lambda).$$

Proof (sketch). The proof follows from the following observation: The map $x \to x^2(-1)^b$ where $b \leftarrow\!\$ \{0,1\}$ sends the uniform distribution on $\mathbb{NR}_N$ to the uniform distribution on $\mathbb{T}_N$, and the uniform distribution on $\mathbb{H}_N \times \mathbb{NR}_N$ to the uniform distribution on $\mathbb{H}_N \times \mathbb{T}_N$. □

Corollary 1.4.3. *Assume that the DCR assumption holds. Then for all PPT adversaries $\mathcal{A}$,*

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(N,x) : x \leftarrow\!\$\ \mathbb{T}_N\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(N,x) : \begin{array}{c} x' \leftarrow\!\$\ \mathbb{T}_N \\ x = x'(1+N) \bmod N^{\xi+1} \end{array}\right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

Proof (sketch). In the first experiment, we replace x with a uniform value over $\mathbb{J}_N$ using the DCR assumption. In the second experiment, we replace x' with a uniform value over $\mathbb{J}_N$ (again using the DCR assumption). We obtain two experiments where x is sampled uniformly over $\mathbb{J}_N$ and thus they are indistinguishable. □

1.4.3 Subgroup Decision

We also use Boneh-Goh-Nissim (BGN) cryptosystem [BGN05] in our range proofs. Thus, its underlying Subgroup Decision (SD) assumption is rephrased as follows for completeness.

Let $\mathcal{G}$ be an algorithm that takes a security parameter as input and outputs $\mathsf{val} := (p, q, \mathbb{G}, \mathbb{G}_1, e)$ such that p, q are primes, $n = pq$ and $\mathbb{G}, \mathbb{G}_1$ are descriptions of groups of order n and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map. Let $q\mathbb{G}$ be the subgroup of $\mathbb{G}$ of order q.

Definition 1.4.3 (Subgroup Decision [GOS06])**.** Let $\mathcal{G}$ be an algorithm that takes a security parameter as input and outputs $\mathsf{val} := (p, q, \mathbb{G}, \mathbb{G}_1, e)$ such that p, q are primes, $n = pq$ and $\mathbb{G}, \mathbb{G}_1$ are descriptions of groups of order n and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map. Let $q\mathbb{G}$ be the subgroup of $\mathbb{G}$ of order q. The *subgroup decision* (SD) assumption holds for generator $\mathcal{G}$ states that for all PPT adversaries $\mathcal{A}$,

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, G, H) : \begin{array}{c} \mathsf{val} \leftarrow \mathcal{G}(1^k), \quad n = pq \\ G, H \leftarrow \mathbb{G}_{gen} \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, G, H) : \begin{array}{c} \mathsf{val} \leftarrow \mathcal{G}(1^k), \quad n = pq \\ G \leftarrow \mathbb{G}_{gen}, \quad H \leftarrow q\mathbb{G} \setminus \{1\} \end{array}\right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

1.4.4 Computational Diffie-Hellman

Definition 1.4.4 (Computational Diffie-Hellman)**.** Let $\mathcal{G}(\lambda)$ be an algorithm that outputs $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is a group of prime order p and g is a generator of the group. The Computational Diffie-Hellman (CDH) assumption holds for generator $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$

$$\Pr\left[g^{a_1 a_2} \leftarrow \mathcal{A}(\mathbb{G}, p, g, g^{a_1}, g^{a_2}) : \begin{array}{c} (\mathbb{G}, p, g) \leftarrow \mathcal{G}(\lambda) \\ a_1, a_2 \leftarrow\!\$\ \mathbb{Z}_p \end{array}\right] \leq \mathsf{negl}(\lambda).$$

1.4.5 Learning with Errors

Definition 1.4.5 (Learning with Errors). Let $q, k \in \mathbb{N}$ where $k \in \mathsf{poly}(\lambda)$, $\mathbf{A} \in \mathbb{Z}_q^{k \times n}$ and $\beta \in \mathbb{R}$. For any $n = \mathsf{poly}(k \log q)$, the Learning with Errors (LWE) assumption holds if for every PPT algorithm $\mathcal{A}$ we have

$$|\Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e})] - \Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{y})]| \leq \mathsf{negl}(\lambda)$$

for $\mathbf{s} \leftarrow\!\!\$\ \{0,1\}^k$, $\mathbf{e} \leftarrow\!\!\$\ D_{\mathbb{Z}^n,\beta}$ and $\mathbf{y} \leftarrow\!\!\$\ \{0,1\}^n$, where $D_{\mathbb{Z}^n,\beta}$ is some error distribution.

Definition 1.4.6 (Learning with Errors (LWE)[Reg05]). For a vector $\mathbf{s} \in \mathbb{Z}_q^n$ called the secret, the LWE distribution $A_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $a \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow\!\!\$\ \chi$, and outputting $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e \mod q)$. Moreover, decisional-$\mathsf{LWE}_{n,m,q,\chi}$ is

$$\begin{aligned} \mathsf{Adv}^{\mathsf{LWE}}_{n,m,q,\chi}(\mathcal{A}) = \big| \Pr[b = 1 | \mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{m \times n}, \mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_q^m;\\ b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{t})]\\ - \Pr[b = 1 | \mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}, \mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n, \mathbf{e} \leftarrow\!\!\$\ \chi^m;\\ b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{As} + \mathbf{e})] \big|. \end{aligned}$$

Definition 1.4.7 (Module Learning With Errors MLWE [BGV12]). For integers m, k, and a probability distribution $D : R_q \to [0,1]$, we say that the advantage of algorithm $\mathcal{A}$ in solving the decisional $\mathsf{MLWE}_{m,k,D}$ problem over the ring R_q is

$$\begin{aligned} \mathsf{Adv}^{\mathsf{MLWE}}_{m,k,D}(\mathcal{A}) = \big| \Pr[b = 1 | \mathbf{A} \leftarrow R_q^{m \times k}, \mathbf{t} \leftarrow R_q^m;\\ b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{t})]\\ - \Pr[b = 1 | \mathbf{A} \leftarrow R_q^{m \times k}, \mathbf{s}_1 \leftarrow D^k, \mathbf{s}_2 \leftarrow D^m;\\ b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{As}_1 + \mathbf{s}_2)] \big| \end{aligned}$$

Definition 1.4.8 (Module Short Integer Solution MSIS[Ajt98]).

$$\begin{aligned} \mathsf{Adv}^{\mathsf{MSIS}}_{m,k,\gamma}(\mathcal{A}) = \Pr\Big[0 < \|\mathbf{y}\|_\infty < \gamma \wedge [\mathbf{I} \,|\, \mathbf{A}] \cdot \mathbf{y} = 0 \,\Big|\\ \mathbf{A} \leftarrow R_q^{m \times k}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A})\Big] \end{aligned}$$

Definition 1.4.9 (The $\mathsf{SelfTargetMSIS}$ Problem in [LDK+20]). Suppose that $\mathsf{H} : \{0,1\}^* \to B_\tau$ is a cryptographic hash function. To an algorithm $\mathcal{A}$ we associate the advantage function

$$\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathsf{H},m,k,\gamma}(\mathcal{A}) = \\ \Pr\left[\begin{array}{l} 0 < \|\mathbf{y}\|_\infty < \gamma \\ \wedge\, H(\mu \,\|\, [\mathbf{I}|\mathbf{A}]\cdot\mathbf{y}) = c \end{array}\middle|\begin{array}{l} \mathbf{A} \leftarrow R_q^{m\times k}; \\ (\mathbf{y} := \begin{bmatrix}\mathbf{r}\\ c\end{bmatrix}, \mu) \leftarrow \mathcal{A}^{|H(\cdot)\rangle}(\mathbf{A}) \end{array}\right]$$

1.4.6 Learning Parity with Noise

The Learning Parity with Noise (LPN) assumption is closely related to the problem of decoding a random linear code. Informally, it states that it is hard to find a solution for a noisy system of linear equations over $\mathbb{Z}_2$.

Definition 1.4.10 (Learning Parity with Noise). Let $n, m, t \in \mathbb{N}$ such that $n \in \mathsf{poly}(\lambda)$ and let $\chi_{m,t}$ be a uniform distribution over the set of error vectors of size m and hamming weight t. The *Learning Parity with Noise* (LPN) assumption $\mathsf{LPN}(n, m, \rho)$ holds if for any PPT adversary $\mathcal{A}$ we have that

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA}+\mathbf{e}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\ \{0,1\}^{n\times m} \\ \mathbf{s} \leftarrow\!\!\$\ \{0,1\}^n \\ \mathbf{e} \leftarrow\!\!\$\ \chi_{m,t} \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{y}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\ \{0,1\}^{n\times m} \\ \mathbf{y} \leftarrow\!\!\$\ \{0,1\}^m \end{array}\right]\right| \le \mathsf{negl}(\lambda)$$

where $\rho = m/t$ (ρ is called the noise rate).

In this thesis, we assume that the noise rate ρ is $m^{1-\varepsilon}$ for any constant $\varepsilon > 0$. The LPN assumption is believed to be hard for that noise rate (see e.g. [BCG+19a] and references therein).

LPN over larger fields. Following [BCG+19a, JLS21], we define the LPN assumption over larger fields $\mathbb{Z}_q$ where $q > 2$ is a prime number. In the following, let $\chi_{m,t,q}$ be the uniform distribution over $\{\mathbf{v} \in \mathbb{Z}_q^m : \mathsf{hw}(\mathbf{v}) = t\}$. In other words, $\chi_{m,t,q}$ is the uniform distribution over the set of vectors in $\mathbb{Z}_q$ which have $m - t$ null coordinates.

Definition 1.4.11 (LPN over larger fields assumption). Let $n, m, t, q \in \mathbb{N}$ such that $n \in \mathsf{poly}(\lambda)$ and q is a prime number, and let $\chi_{m,t,q}$ be as above. The *LPN over larger fields* assumption $\mathsf{LPN}(n, m, \rho, q)$ holds if for any PPT adversary $\mathcal{A}$ we have that

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA}+\mathbf{e}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m} \\ \mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n \\ \mathbf{e} \leftarrow\!\!\$\ \chi_{m,t,q} \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{y}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m} \\ \mathbf{y} \leftarrow\!\!\$\ \mathbb{Z}_q^m \end{array}\right]\right| \le \mathsf{negl}(\lambda)$$

where $\rho = m/t$.

1.5 Threshold Public-key Encryption

We present some ideal functionalities regarding threshold public-key encryption (TPKE) schemes. In the following, N is the number of parties.

Let $\mathcal{F}_{\mathsf{Gen}}$ be the ideal functionality that distributes a secret share of the secret key and the corresponding public key. That is, on input $(\mathsf{sid}, \mathsf{P}_i)$, $\mathcal{F}_{\mathsf{Gen}}$ outputs $(\mathsf{pk}, \mathsf{sk}_i)$ to each party party where $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TPKE.Gen}(1^\lambda, N)$.

Moreover, we define the functionality $\mathcal{F}_{\mathsf{DecZero}}$, which allows N parties, each of them holding a secret share sk_i, to learn if a ciphertext is an encryption of 0 and nothing else. That is, $\mathcal{F}_{\mathsf{DecZero}}$ receives as input a ciphertext c and the secret shares of each of the parties. It outputs 0, if $0 \leftarrow \mathsf{Dec}(\mathsf{sk}_, \ldots \mathsf{Dec}(\mathsf{sk}_N, c) \ldots)$, and 1 otherwise. Note that these functionalities can be securely realized on various PKE schemes such as El Gamal PKE or Pailler[5] PKE [HV17].

We also assume that the underlying TPKE (or plain PKE) is always additively homomorphic unless stated otherwise.

Definition 1.5.1 (Threshold Public-Key Encryption)**.** A Threshold Public-Key Encryption (TPKE) scheme is defined by the following algorithms:

- $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}(1^\lambda, N)$ takes as input a security parameter. It outputs a public key pk and N secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ takes as input a public key pk and a message $m \in \{0,1\}^*$. It outputs a ciphertext c.
- $c' \leftarrow \mathsf{Dec}(\mathsf{sk}_i, c)$ takes as input one of the secret keys sk_i and a ciphertext. It outputs a share decryption c' of c.

Correctness. For any $N \in \mathbb{N}$ and any permutation $\pi : [N] \to [N]$, we have that

$$\Pr\left[m \leftarrow \mathsf{Dec}(\mathsf{sk}_{\pi(N)}, \mathsf{Dec}(\mathsf{sk}_{\pi(N-1)}, \ldots \mathsf{Dec}(\mathsf{sk}_{\pi(1)}, \mathsf{Enc}(\mathsf{pk}, m)) \ldots))\right] = 1$$

where $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}(1^\lambda, N)$.

IND-CPA security. For any $N \in \mathbb{N}$, any permutation $\pi : [N] \to [N]$ and any adversary $\mathcal{A}$, we require that

$$\Pr\left[b \leftarrow \mathcal{A}(c, \mathsf{st}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}(1^\lambda, N) \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}\left(\mathsf{pk}, \mathsf{sk}_{\pi(1)}, \ldots, \mathsf{sk}_{\pi(k)}\right) \\ b \leftarrow\!\!\$\ \{0,1\} \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

[5] We will assume the message space of Paillier's cryptosystem as a field as also mentioned in [KMWF07].

for any $k < N$.

Additive Homomorphism. We also assume that the TPKE (or plain PKE) is homomorphic for additive operation.[6] That is, for all $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}(1^\lambda, N)$, we can define two groups $(\mathcal{M}, \oplus), (\mathcal{C}, \otimes)$ such that, given two ciphertexts $c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1)$ and $c_2 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_2)$, we require that

$$c_1 \otimes c_2 = \mathsf{Enc}(\mathsf{pk}, m_1 \oplus m_2).$$

By abuse of notation, we usually denote the operations of $\mathcal{M}$ and $\mathcal{C}$ as $+$.

1.6 Programmable Pseudorandom Functions

Pseudorandom functions (PRF) are ubiquitous objects in cryptography. We present the definition of PRF in the following.

Definition 1.6.1 (Pseudorandom Function). A *Pseudorandom Function* (PRF) is defined by a keyed function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that, for any PPT adversary $\mathcal{A}$

$$|\Pr[1 \leftarrow \mathcal{A}(y, x) : y \leftarrow \mathsf{PRF}(k, x)] - \Pr[1 \leftarrow \mathcal{A}(y, x) : y \leftarrow f(x)]| \leq \mathsf{negl}(\lambda)$$

for any $x \in \mathcal{X}$, where $f : \mathcal{X} \to \mathcal{Y}$ is a uniformly chosen random function and the key k is sampled uniformly at random from $\mathcal{K}$.

A programmable PRF allows the simulator to program the output of a PRF on several inputs at key generation time.

Definition 1.6.2 (Programmable PRF [KMP+17]). A programmable PRF (PPRF) is composed of the following algorithms:

- $k = (k', \mathsf{hint}) \leftarrow \mathsf{KeyGen}(1^\lambda, (x, y))$ takes as input a security parameter and a pair of points $(x, y) \in \mathcal{X} \times \mathcal{Y}$. It outputs a key k' and a hint hint.
- $y \leftarrow \mathsf{PPRF}(k, x)$ takes as input a key $k \in \mathcal{K}$ and a value $x \in \mathcal{X}$. It outputs $y \in \mathcal{Y}$.

Correctness of the PPRF states that $y \leftarrow \mathsf{PPRF}(k, x)$ for the programmed point (x, y). Security roughly states that it is hard for the adversary to guess the point x which was programmed even given the hint (see [KMP+17]).

[6] From now on, we always assume that PKE and TPKE used in this thesis fulfill this property, unless stated otherwise.

An example. Let $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \{0,1\}^\ell$ and $\mathsf{Primes}(\ell)$ be the primes of length ℓ. In this thesis, we use a programmable PRF $\mathsf{PPRF} : \mathcal{K} \times (\mathcal{X} \times \mathbb{Z}) \to \mathsf{Primes}(\ell)$ in which the key (and the hint) is of the form $K = (k, k' = (k'_1, \ldots, k'_\xi)) \in \mathcal{K} \times \{0,1\}^{\ell\xi}$ and where the output of an element $x \in \mathcal{X}$ is computed as i) Start by initializing $i = 1$. ii) Compute $y = \mathsf{PPRF}(k, (x, i)) \oplus k'_i$. iii) Output y, if it is a prime number; else, set $i = i + 1$ and return to step ii); repeat until $i = \xi$. It is easy to see that, under standard number-theoretic assumptions, the process described above outputs a prime number after $\mathcal{O}(\log 2^\ell)$ steps (e.g., [FT14]). If we set $\xi \in \mathcal{O}((\log 2^\ell)^2)$, a direct calculation yields that the probability of not existing any $i \in [\xi]$ such that $\mathsf{PPRF}(k, (x, i)) \oplus k'_i$ is not a prime is negligible in ℓ.

In order to program the output of PPRF at some input x, we first sample a prime number p and an index i from a suitable distribution.[7] Then, we set $k'_i = p \oplus \mathsf{PPRF}(k, (x, i))$. Finally, we choose k'_j, for all $j < i$, uniformly at random such that $\mathsf{PPRF}(k, (x, j)) \oplus k'_j$ is not a prime number. All other k'_j, for $j > i$ are chosen uniformly at random. Such a procedure will succeed with non-negligible probability.

This is a special case of the PPRF designed in [KMP+17] and it is easy to see that, if the PPRF is programmed on a pair of points $(x, y) \in \mathcal{X} \times \mathsf{Primes}(\ell)$ where $y \leftarrow\!\!\$\ \mathsf{Primes}(\ell)$, then it is hard for any PPT adversary $\mathcal{A}$ to guess the programmed point x.

A remark. We slightly overload the notation and denote k as the PPRF key (which is composed of a PRF key k' and a hint hint as in Definition 1.6.2). We do this because, in our case, the hint (when it is a uniformly random value) reveals nothing about the programmed value [KMP+17]. That is, we will use the notation $K \leftarrow \mathsf{KeyGen}(1^\lambda, (x, y))$ where $K = (k, k' = \mathsf{hint})$.

1.7 Puncturable Pseudorandom Functions

In this section, we recall another variant of PRF as follows. Puncturable pseudorandom functions (PPRFs) [BW13, KPTZ13, BGI14] are a special case of PRFs where a punctured key allows one to evaluate the PRF at all points except one.

Definition 1.7.1 (Puncturable PRF)**.** Let $\alpha = \alpha(\lambda)$ and $\beta = \beta(\lambda)$ be two polynomials. A puncturable PRF (PPRF) scheme $\mathsf{PPRF}_{\alpha,\beta} = \mathsf{PPRF}$ is composed by the following algorithms:

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

[7] The index i is sampled from the distribution of the number of uniform samples we need to perform in order to find a prime number. Such a distribution can be easily simulated by just running a prime sampler with true randomness and output i (the number of trials until success) instead of the prime.

Definition 1.7.2 (Correctness)**.** A PPRF scheme PPRF is said to be correct if for all $\lambda \in \mathbb{N}$, for all $S \subseteq (\{0,1\}^{\alpha})^t$ (for $t = \mathsf{poly}(\lambda)$), all $\mathbf{x} \notin S$ we have that

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

Definition 1.7.3 (Pseudorandomness)**.** A PPRF scheme PPRF is said to be pseudorandom at punctured points if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have that

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

PPRFs can be built solely based on any length-doubly pseudorandom generators (PRG)[8] via (a variant of) the tree-based construction of [GGM86]. Throughout this thesis, we call the term GGM-PPRF to this scheme and denote it by $\mathsf{PPRF}_{\mathsf{GGM}}$.

1.8 Designated-Verifier Non-Interactive Zero-Knowledge

NIZK is a cryptographic primitive that allows a prover to prove that it holds a witness for a certain NP statement to a verifier in just one message. In the designated-verifier setting, only a designated party can verify the validity of proofs. This is in contrast with standard NIZK where the verification algorithm can be run by any party.

Let $\mathcal{Z}$ be the set of statements and $\mathcal{W}$ be the set of witnesses. Let $\mathcal{L}$ be a NP language with relation $\mathcal{R}$ such that $z \in \mathcal{L}$ if there is a $w \in \mathcal{W}$ such that $\mathcal{R}(z, w) = 1$.

Definition 1.8.1 (DV-NIZK)**.** Let $\mathcal{L}$ be a NP language. A *Designated-Verifier Non-Interactive Zero-Knowledge* (DV-NIZK) for language $\mathcal{L}$ is composed by the following algorithms:

- $\mathsf{GenCRS}_{\mathcal{L}}(1^\lambda)$ takes as input a security parameter. It outputs a common reference string crs together with the corresponding trapdoor td.

- $\mathsf{Prove}_{\mathcal{L}}(\mathsf{crs}, x, w)$ takes as input a common reference string crs, a statement x and a witness w. It outputs a proof π.

- $\mathsf{Verify}_{\mathcal{L}}(\mathsf{td}, x, \pi)$ takes as input a common reference string crs, a trapdoor td, a statement x and a proof π. It outputs a bit $b \in \{0,1\}$.

A DV-NIZK should fulfill the following properties: completeness, soundness and honest-verifier zero-knowledge.

[8]Which in turn, can be based on LWE, DDH or QR assumptions.

- **Completeness:** A DV-NIZK is *correct* if for all pairs (x, w) such that $\mathcal{R}(x, w) = 1$,

$$\Pr\left[1 \leftarrow \mathsf{Verify}_{\mathcal{L}}(\mathsf{td}, x, \pi) : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_{\mathcal{L}}(1^{\lambda}) \\ \pi \leftarrow \mathsf{Prove}_{\mathcal{L}}(\mathsf{crs}, x, w) \end{array}\right] = 1.$$

- **Statistical Reusable Soundness:** A DV-NIZK is *statistical reusable sound* if for all computationally unbounded adversaries $\mathcal{A}$ and all $x \notin \mathcal{L}$,

$$\Pr\left[1 \leftarrow \mathsf{Verify}_{\mathcal{L}}(\mathsf{td}, x, \pi) : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_{\mathcal{L}}(1^{\lambda}) \\ \pi \leftarrow \mathcal{A}^{\mathsf{Verify}_{\mathcal{L}}(\mathsf{td}, \cdot, \cdot)}(\mathsf{crs}, x) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

  Remark that, in the statistical setting, selective soundness is equivalent to adaptive soundness.

- **Zero-knowledge:** A DV-NIZK is said to be *zero-knowledge* if for all adversaries $\mathcal{A}$ there is an simulator Sim such that

$$\left|\begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}, x, \pi) : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_{\mathcal{L}}(1^{\lambda}) \\ \pi \leftarrow \mathsf{Prove}_{\mathcal{L}}(\mathsf{crs}, x, w) \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}, x, \pi) : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_{\mathcal{L}}(1^{\lambda}) \\ \pi \leftarrow \mathsf{Sim}_{\mathcal{L}}(\mathsf{td}, x) \end{array}\right] \end{array}\right| \leq \mathsf{negl}(\lambda).$$

  When $\mathcal{A}$ is computationally bounded, we say that zero knowledge holds computationally. When $\mathcal{A}$ is computationally unbounded, if its advantage is negligible in the security parameter, we say that zero-knowledge holds statistically while if its advantage is zero, then zero-knowledge holds perfectly.

Range Proof Systems for DJ Ciphertexts. In this thesis, we construct a range-proof system for DJ ciphertexts. That is, we build a DV-NIZK scheme that allows the prover to prove that a given DJ ciphertext ct encrypts a message $m \in [-B, B]$ for some public $B \in \mathbb{Z}$.

Such a scheme can be constructed in the random oracle model (ROM) using the Fiat-Shamir transform (e.g., [DJ01, BBC+18, BBB+18, TBM+20] just to name a few). However, we focus on efficient range proofs in the standard model in this thesis.

1.9 Private Information Retrieval

Private Information Retrieval (PIR) schemes[CGKS95] allow a user to retrieve the i-th bit of an n-bit database, without revealing to the database holder the value of i. Besides, we require an additional privacy property in our schemes: sender privacy (or data privacy)[DMO00].

Definition 1.9.1 (PIR). A private information retrieval (PIR) scheme PIR is composed by the following algorithms:

- $\mathsf{Query}(n, i)$ takes as input an index $i \in [n]$. It outputs a query q and a state st_i.

- $\mathsf{Send}(DB, \mathsf{q})$ takes as input a database $DB \in \{0,1\}^n$ and a message q. It outputs a response r.
- $\mathsf{Retrieve}(\mathsf{r}, \mathsf{st}_i)$ takes as input a response r and a state st_i. It retrieves the entry DB_i.

Definition 1.9.2 (Correctness)**.** A PIR scheme PIR is said to be correct if for any $n \in \mathbb{N}$, $DB \in \{0,1\}^n$ and $i \in [n]$, we have that

$$\Pr\left[DB_i = \mathsf{Retrieve}(\mathsf{st_i}, \mathsf{r}) : \begin{array}{c} (\mathsf{st_i}, \mathsf{q}) \leftarrow \mathsf{Query}(n, i) \\ \mathsf{r} \leftarrow \mathsf{Send}(DB, \mathsf{q}) \end{array}\right] = 1.$$

Definition 1.9.3 (User privacy)**.** A PIR scheme PIR is said to be user private if for any PPT adversary $\mathcal{A}$, any $n, \lambda \in \mathbb{N}$, $DB \in \{0,1\}^n$ and $i, j \in [n]$, we have that

$$\left| \begin{array}{c} \Pr[1 \leftarrow \mathcal{A}(1^\lambda, DB, \mathsf{q_i}) : (\mathsf{st_i}, \mathsf{q_i}) \leftarrow \mathsf{Query(n, i)}] - \\ \Pr[1 \leftarrow \mathcal{A}(1^\lambda, DB, \mathsf{q_j}) : (\mathsf{st_j}, \mathsf{q_j}) \leftarrow \mathsf{Query(n, j)}] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

Definition 1.9.4 (Sender privacy)**.** A PIR scheme PIR is said to be sender private if for any $\lambda \in \mathbb{N}$, any $n = \mathsf{poly}(\lambda)$, any $i \in [n]$ and any two databases $DB^x, DB^y \in \{0,1\}^n$ such that $DB_i^x = DB_i^y$ we have that for all PPT adversaries $\mathcal{A}$

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, i, n, \mathsf{st}_i, \mathsf{r}_i) : \begin{array}{c} (\mathsf{st}_i, \mathsf{q}_i) \leftarrow \mathsf{Query}(n, i) \\ \mathsf{r}_i \leftarrow \mathsf{Send}(DB^x, \mathsf{q}_i) \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, i, n, \mathsf{st}_i, \mathsf{r}_i) : \begin{array}{c} (\mathsf{st}_i, \mathsf{q}_i) \leftarrow \mathsf{Query}(n, i) \\ \mathsf{r}_i \leftarrow \mathsf{Send}(DB^y, \mathsf{q}_i) \end{array}\right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

Black-box constructions for PIR exist LWE, DDH or QR assumptions [DGI+19].

2
Threshold Private Set Intersection

In this chapter, we begin by addressing the first problem in privacy-preserving computation, known as threshold private set intersection (tPSI) in a multiparty setting. Our investigation focuses on the communication bandwidth of tPSI, and we present improvements to its asymptotic performance.

To recap, threshold private set intersection enables multiple parties to calculate the intersection of their input sets, provided that the intersection is larger than $n - t$, where n represents the size of each set and t is a predetermined threshold. The primary advantage of this primitive is that, unlike standard private set intersection (PSI), the established upper bounds on communication complexity depend solely on the threshold t and not on the input sets' sizes.

Current tPSI protocols are divided into two components: A cardinality testing phase, where parties determine if the intersection is larger than a certain threshold; And a PSI phase, where the actual intersection is computed. The primary source of inefficiency in threshold PSI lies in the former component.

In this chapter, we introduce a new cardinality testing protocol that enables N parties to verify whether the intersection of their input sets is larger than $n - t$. The protocol results in a communication complexity of $\tilde{\mathcal{O}}(Nt^2)$. Consequently, we obtain a threshold PSI scheme for N parties with a communication complexity of $\tilde{\mathcal{O}}(Nt^2)$.

2.1 Overview

We first recall the definition of PSI as follows. Suppose Alice holds a set S_A and Bob a set S_B. Private set intersection is a cryptographic primitive that allows each party to learn the intersection $S_A \cap S_B$ and nothing else. In particular, Alice gets no information about $S_B \setminus S_A$ (and vice-versa). The problem has attracted a lot of attention through the years, with an extended line of work proposing solutions in a variety of different settings (e.g., [Mea86, FNP04, KS05, DMRY09, DKT10, DCW13, PSZ14, PSSZ15, KKRT16, RR17a, HV17, RR17b, PSWW18, GN19, GS19a, PRTY19]). Also, numerous applications have been proposed for PSI such as contact discovery, advertising, etc (see for example [IKN$^+$17] and references therein). More recently, PSI has also been proposed as a solution for private contact tracing (e.g., [BBV$^+$20]).

Threshold PSI. In this chapter, we focus on a special set of PSI called *Threshold PSI*. Here, the parties involved in the protocol learn the output if the size of the intersection between the input sets of the parties is very large, say larger than $n - t$, where n is the size of the input sets and t is some *threshold* such that $t \ll n$; Otherwise, they learn nothing about the intersection. This is in contrast with standard PSI where the parties always get the intersection, no matter its size.

The main reason for considering this problem (apart from its numerous applications which we discuss next) is that the amount of communication needed is much smaller than for standard PSI: In particular, there are threshold PSI protocols whose communication complexity depends only on the threshold t and not on the size of the input sets as for standard PSI [GS19a].

Despite its theoretical and practical appeal, there are just a few works that consider this problem [HOS17, GN19, GS19a], and just one of them achieves communication complexity independent of n [GS19a], in the two-party setting.

2.1.1 Applications of Threshold PSI

A wide number of applications have been suggested for threshold PSI in previous works such as applications for dating apps or biometric authentication mechanisms [GS19a].

One of the most interesting applications for threshold PSI is its use in carpooling (or ridesharing) apps. Suppose two (or more) parties are using a carpooling app, which allows them to share a vehicle if their routes have a large intersection. However, due to privacy issues, they do not want to make their itinerary public. Threshold PSI solves this problem in a simple way [HOS17]: The parties can engage in a threshold PSI protocol, learn the intersection of the routes and, if the intersection is large enough, share a vehicle. Otherwise, they learn nothing and their privacy is maintained.

PSI using Threshold PSI. As we mentioned before, most of the current protocols for threshold PSI (including ours) are split into two parts: i) A *cardinality testing*, where parties decide if the intersection is larger than $n - t$; And ii) secure computation of the intersection of the input sets (which we refer to as the PSI part). The communication complexity of these two parts should depend only on the threshold t and not on the input sets' size n.

Threshold PSI protocols of this form can be used to efficiently compute the intersection, even when *no threshold* on the intersection is known a priori by the parties, by doing an exponential search for the *right* threshold. In this case, parties can proceed as follows:

1. Run a cardinality testing for some t (say $t = 1$).
2. If it succeeds, perform the PSI part. Else, run again the cardinality test for $t = 2t$.

3. Repeat Step 2 until the cardinality testing succeeds for some threshold t and the set intersection is computed.

By following this blueprint, parties are sure that they overshoot the right threshold by a factor of at most 2. That is if the intersection is larger than $n - t'$, then the cardinality testing will succeed for t such that $t \geq t' > t/2$. Thus, they can compute the intersection incurring only in a factor of 2 overhead over the best insecure protocol. In other words, PSI protocols can be computed with communication complexity depending on the size of the intersection, and not on the size of the sets.

This approach can be useful in scenarios where parties suspect that the intersection is large but they do not know exactly how large it is.

2.1.2 Contributions

In the following discussion, N represents the number of parties participating in a multiparty protocol, while t refers to the threshold in a threshold PSI protocol. Here, we provide a concise overview of our results.

Multi-party Cardinality Testing. We develop a new cardinality testing scheme that allows N parties to check if the intersection of their input sets, each having size n, is larger than $n - t$ for some threshold $t \ll n$. The protocol needs $\tilde{\mathcal{O}}(Nt^2)$ bits of information to be exchanged.

Along the way, we develop new protocols to securely compute linear algebra-related functions (such as computing the rank of an encrypted matrix, inverting an encrypted matrix or even solving an encrypted linear system). Our protocols build on ideas of previous works [NW06, KMWF07], except that our protocols are specially crafted for the multi-party case. Technically, we rely heavily on Threshold Public-Key Encryption schemes which are additively homomorphic (such schemes can be constructed from DDH [Elg85], DCR [Pai99], or from several pairings assumptions [BBS04, BGN05]) to perform linear operations.

Multi-party Threshold PSI. We then show how our cardinality testing protocol can be used to build a Threshold PSI protocol in the multi-party setting. Our construction achieves communication complexity of $\tilde{\mathcal{O}}(Nt^2)$.

Concurrent Work

Recently, Ghosh and Simkin [GS19b] updated their paper with a generalization to the multi-party case which is similar to the one presented in this paper in Section 2.5. However, they leave as a major open problem the design of a new Cardinality Testing that extends nicely to multiple parties, a problem on which we make relevant advances in this work.

In a concurrent work, Badrinarayanan *et al.* [BMRR21] also proposed new protocols for threshold PSI in the multi-party setting. Their results complement ours. In particular, they propose an FHE-based approach to solve the same problem as we do with a communication complexity of $\mathcal{O}(Nt)$, where N is the number of parties and t is the threshold. However, we remark that the goal of our work was to reduce the assumptions needed for threshold PSI. They also propose a TPKE-based protocol that solves a slightly different problem: the parties learn the intersection if and only if the difference between the union and the intersection is small, that is, $|\left(\cup_{i=1}^{N} S_i\right) \setminus \left(\cap_{i=1}^{N} S_i\right)|$ is small[1], which is denoted as $\mathcal{F}_{\mathsf{TPSI\text{-}diff}}$ in [BMRR21]. This protocol achieves communication complexity of $\tilde{\mathcal{O}}(Nt)$. They achieve that result using completely different techniques from the ones used in this work. Namely, they noticed that computing the determinant of a Hankel matrix can be done in sublinear time in the size of the matrix. This implies that the cardinality testing of [GS19a] can actually be realized in time $\tilde{\mathcal{O}}(Nt)$.

2.2 Techniques

We now give a high-level overview of the techniques we use to achieve the results discussed above.

Threshold PSI: The Protocol of [GS19a]

Consider two parties Alice and Bob, with their respective input, sets S_A and S_B of size n. Suppose that they want to know the intersection $S_A \cap S_B$ iff $|S_A \cap S_B| \geq n - t$ for some threshold $t \ll n$. To compute the intersection, both parties encode their sets into polynomials $P_A(x) = \prod_i^n (x - a_i)$ and $P_B(x) = \prod_i^n (x - b_i)$ over a large finite field $\mathbb{F}$, where $a_i \in S_A$ and $b_i \in S_B$. The main observation of Ghosh and Simkin [GS19a] is that *set reconciliation techniques* (developed by Minsky et al. [MTZ03]) can be applied in this scenario: if $|S_A \cap S_B| \geq n - t$, then

$$\frac{P_A(x)}{P_B(x)} = \frac{P_{A\cap B}(x)}{P_{A\cap B}(x)} \frac{P_{A\setminus B}(x)}{P_{B\setminus A}(x)} = \frac{P_{A\setminus B}(x)}{P_{B\setminus A}(x)}$$

and, moreover, $\deg P_{A\setminus B} = \deg P_{B\setminus A} = t$. Hence, Alice and Bob just need to (securely) compute $\mathcal{O}(t)$ evaluation points of the rational function $P_A(x)/P_B(x) = P_{A\setminus B}(x)/P_{B\setminus A}(x)$ and, after interpolating over these points, Bob can recover the denominator (which reveals the intersection).

Of course, Bob should not be able to recover the numerator $P_{A\setminus B}$, otherwise, security is compromised. So, [GS19a] used an Oblivious Linear Evaluation (OLE) scheme to *mask* the numerator with a random polynomial that hides $P_{A\setminus B}$ from Bob.

This protocol is only secure if Alice and Bob are absolutely sure that $|S_A \cap S_B| \geq n - t$. Otherwise, additional information could be leaked about the respective inputs. Consequently, Alice and Bob should perform a *cardinality testing* protocol, which reveals if $|S_A \cap S_B| \geq n - t$ and nothing else.

Limitations of the protocol when extending to the multi-party setting. It turns out that the main source of inefficiency when extending the Ghosh and Simkin protocol to the multi-

[1] It is a slightly different problem from the one we solve in this work. Here, we want to disclosure the intersection $\cap_{i=1}^{N} S_i$ if $|\cap_{i=1}^{N} S_i| \geq n - t$, which is denoted as $\mathcal{F}_{\mathsf{TPSI\text{-}int}}$ in [BMRR21].

party setting is the cardinality testing they use. In [GS19a], Alice and Bob encode their sets into polynomials $Q_A(X) = \sum_i^n x^{a_i}$ and $Q_B(X) = \sum_i^n x^{b_i}$, respectively, where $a_i \in S_A$ and $b_i \in S_B$. Then, they can check if $\tilde{Q}(x) = Q_A(x) - Q_B(x)$ is a *sparse* polynomial. If it is, we conclude that the set $(S_A \cup S_B) \setminus (S_A \cap S_B)$ is small. By disposing of $\mathcal{O}(t)$ evaluations of the polynomial $\tilde{Q}(x)$ in a Hankel matrix [GJR10] and securely computing its determinant (via a generic secure linear algebra protocol from [KMWF07]), both parties can determine if $|S_A \cap S_B| \geq n - t$. The total communication complexity of this protocol is $\mathcal{O}(t^2)$.[2]

However, if we were to *naively* extend this approach to the multi-party setting, we would have N parties computing, say,

$$\tilde{Q}(x) = NQ_1(x) - Q_2(x) - \cdots - Q_N(x)$$

which is a sparse polynomial only if N is small. Moreover, if we were to compute the sparsity of this polynomial using the same approach, we would have a protocol with communication complexity $\mathcal{O}((Nt)^2)$.

Our Approach

Given the state of affairs presented in the previous section, it seems we need to take a different approach from the one of [GS19a] if we want to design an efficient threshold PSI protocol for multiple parties.

Interlude: Secure Linear Algebra. Recall that in the setting of *secure linear algebra* (as in [NW06] and [KMWF07]), there are two parties, one holding encryption of a matrix $\mathsf{Enc}(\mathsf{pk}, \mathbf{M})$ and the other one holding the corresponding secret key sk. Their goal is to compute an encryption of a (linear algebra-related) function of the matrix $\mathbf{M}$, such as the rank, the determinant of $\mathbf{M}$, or, most importantly, find a solution $\mathbf{x}$ for the linear system $\mathbf{Mx} = \mathbf{y}$ where both $\mathbf{M}$ and $\mathbf{y}$ are encrypted. We can easily extend this problem to the multi-party case: Consider N parties, $\mathsf{P}_1, \ldots, \mathsf{P}_N$, each one holding a share of the secret key of a threshold PKE scheme. Additionally, P_1 has an encrypted matrix. The goal of all the parties is to compute an encryption of a (linear algebra-related) function of the encrypted matrix.

We observe that the protocols for secure linear algebra presented in [KMWF07] can be extended to the multiparty setting by replacing the use of an (additively homomorphic) PKE and garbled circuits for an (additively homomorphic) threshold PKE[3]. Hence, our protocols allow N parties to solve a linear system of the form $\mathbf{Mx} = \mathbf{y}$ under the hood of a threshold PKE scheme.

[2]Given this, we conclude that the communication complexity of the threshold PSI protocol of [GS19a] is dominated by this cardinality testing protocol.

[3]We need a bit-conversion protocol such as [ST06] to convert between binary circuits and algebra operations.

Cardinality Testing via Degree Test of a Rational Function. Consider again the encodings $P_{S_i}(x) = \prod_j^n (x - a_j^{(i)})$ where $a_j^{(i)} \in S_i$, for N different sets, and the rational function[4]

$$\frac{P_{S_1} + \cdots + P_{S_N}}{P_{S_1}} = \frac{P_{S_1 \setminus (\cap_{j=1}^N S_j)} + \cdots + P_{S_N \setminus (\cap_{j=1}^N S_j)}}{P_{S_1 \setminus (\cap_{j=1}^N S_j)}}.$$

Note that, if the intersection $\cap S_i$ is larger than $n-t$, then $\deg P_{S_1 \setminus (\cap_{j=1}^N S_j)} = \cdots = \deg P_{S_N \setminus (\cap_{j=1}^N S_j)} \leq t$.

Therefore, the cardinality testing boils down to the following problem: Given a rational function $f(x) = \tilde{P}_1(x)/\tilde{P}_2(x)$, can we securely decide if $\deg \tilde{P}_1 = \deg \tilde{P}_2 \leq t$ having access to $\mathcal{O}(t)$ evaluation points of $f(x)$?

Our crucial observation is that, if we interpolate two different rational functions f_V and f_W on different two support sets $V = \{v_i, f(v_i)\}$ and $W = \{w_i, f(w_i)\}$ each one of size $2t$, then we have:

1. $f_V = f_W$ if $\deg P_1 = \deg P_2 \leq t$
2. $f_V \neq f_W$ if $\deg P_1 = \deg P_2 > t$

except with negligible probability over the uniform choice of v_i, w_i.

Moreover, interpolating a rational function can be reduced to solving a linear system of equations. Hence, by using the Secure Linear Algebra tools developed before, we can perform the *degree test* revealing nothing else than the output. In other words, we can decide if the size of the intersection is smaller than $n - t$ while revealing no additional information about the parties' input sets.

Security of the protocol. We prove the security of our cardinality testing in the UC framework [Can01]. However, there is a subtle issue with our security proof. Namely, our secure linear algebra protocols cannot be proven UC-secure since the inputs are encrypted under a public key which, in the UC setting, needs to come from somewhere.

We solve this problem by using the Externalized UC framework [CDPW07]. In this framework, the secure linear algebra ideal functionalities all share a common setup which, in our case, is the public key (and the corresponding secret key shares). We prove the security of our secure linear algebra protocols in this setting.

Since the secure linear algebra protocols are secure if they all share the same public key, then, on the cardinality testing, we just need to create this public key and share it over these functionalities. Thus, we prove the standard UC-security of our cardinality testing.

Badrinarayanan et al. [BMRR21] also encounter the same problem as we did and they opted to not prove the security of each subprotocol individually, but rather prove security only for their main protocol (where the public key is created and shared among these smaller protocols).

[4]We actually need to randomize the polynomials in the numerator to guarantee correctness, that is, we need to multiply each term in the numerator by a uniformly chosen element. This is in contrast with the two-party setting where correctness holds even without randomizing the numerator. However, we omit this step for simplicity.

Multi-party PSI. Having developed cardinality testing, we can now focus on securely computing the intersection. In fact, our protocol for computing the intersection can be seen as a *generalization* of Gosh and Simkin protocol [GS19a]. Again, by encoding the sets as above (that is, $P_{S_i}(x) = \prod_j^n (x - a_j^{(i)})$ where $a_j^{(i)} \in S_j$ and S_j is the set of party P_j) and knowing that the intersection is larger than $n - t$, parties can securely compute the rational function[5] $(P_{S_1} + \cdots + P_{S_N})/P_{S_1}$. By interpolating the rational function on any $\mathcal{O}(t)$ points, party P_1 can recover the denominator and compute the intersection.

The main difference between our protocol and the one in [GS19a] is that we replace the OLE calls used in [GS19a] with a threshold additively homomorphic PKE scheme (which can be seen as the multi-party replacement of OLE).

2.2.1 Other Related Work

Oblivious Linear Algebra. Cramer and Damgård [CD01] introduced a constant-round protocol to securely solve linear systems of unknown rank over a finite field. Although their main focus was on round optimality, their proposal's communication cost is $\Omega(t^3)$ for an input size of $\mathcal{O}(t^2)$. Bouman et al. [BdV18] recently developed a secure linear algebra protocol for multiple parties, with their focus being on computational complexity.

Other secure linear algebra schemes in the two-party setting have been presented by Nissim and Weinreb in [NW06] and Kiltz et al. in [KMWF07]. In the following, we consider (square) matrices of size t over a field $\mathbb{F}$. These two works employ different approaches: [NW06] addresses linear algebra-related problems obliviously via Gaussian elimination, resulting in an $\mathcal{O}(t^2)$ communication complexity for a square matrix of size t. However, their approach has an error probability that decreases polynomially with t, meaning that the error probability is only sufficiently small when applied to a linear system with large matrices. On the other hand, [KMWF07] has an error probability that decreases polynomially with $|\mathbb{F}|$, making it negligible when $\mathbb{F}$ is of exponential size.[6]

2.3 Definitions

Multi-Party Threshold Private Set Intersection. This ideal functionality implements the multi-party version of the functionality above. Here, each of the N parties inputs a set and they learn the intersection if and only if the intersection is large enough.

[5]Again, we omit the randomization of the polynomials. Actually, without randomization, these methods (including [GS19a]) are exactly the same as the technique for the set reconciliation problem in [MTZ03].

[6]This is important to us since, in the threshold PSI setting, $t \ll n$ where t is the threshold and n is the set size. Kiltz et al. solve linear algebra problems via minimal polynomials, and use adaptors between garbled circuits and additive homomorphic encryption to reduce round complexity. In this work, we extend Kiltz's protocol to the multiparty case without using garbled circuits (otherwise the circuit size would depend on the number of parties) while preserving the same communication complexity for each party ($\mathcal{O}(t^2)$).

$\mathcal{F}_{\mathsf{MTPSI}}$ functionality

PARAMETERS: $\mathsf{sid}, N, t \in \mathbb{N}$ known to both parties.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_i, S_i)$ from party P_i, $\mathcal{F}_{\mathsf{MTPSI}}$ stores S_i and ignores future messages from P_i with the same sid.
- Once $\mathcal{F}_{\mathsf{MTPSI}}$ has stored all inputs S_i, for $i \in [n]$, it does the following: If $|S_1 \setminus (\cap_{i=2}^{N} S_i)| \leq t$, $\mathcal{F}_{\mathsf{MTPSI}}$ outputs $S_\cap = \cap_{i=1}^{N} S_i$. Else, it outputs $\perp$.

Externalized UC Protocol with Global Setup

We introduce a notion of protocol emulation from [CDPW07], called externalized UC emulation (EUC), which is a simplified version of UC with a global setup (GUC).

Definition 2.3.1 (EUC-Emulation [CDPW07]). We say that π EUC-realizes $\mathcal{F}$ with respect to shared functionality $\bar{\mathcal{G}}$ (or, in shorthand, that π $\bar{\mathcal{G}}$-EUC-emulates φ) if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary Sim such that for any shared functionality $\bar{\mathcal{G}}$, we have:

$$\mathsf{IDEAL}^{\bar{\mathcal{G}}}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}} \approx \mathsf{EXEC}^{\bar{\mathcal{G}}}_{\pi,\mathcal{A},\mathcal{Z}}$$

Notice that the formalism implies that the shared functionality $\bar{\mathcal{G}}$ exists both in the model for executing π and also in the model for executing the ideal protocol for $\mathcal{F}$, $\mathsf{IDEAL}_{\mathcal{F}}$.

We remark that the notion of $\bar{\mathcal{G}}$-EUC-emulation can be naturally extended to protocols that use several different shared functionalities (instead of only one).

Throughout this work, φ will denote the Euler's totient function.

Let $\Phi_{\mathbb{Z},\beta}$ be the distribution that outputs a uniformly chosen value in $\mathbb{Z}$ from the interval $[-\beta, \beta]$. We call *shifted rectangle* to this distribution [AIK11]. The following lemma states that we can *drown* (i.e., statistically hide) a value using a sample from a much wider $\Phi_{\mathbb{Z},\beta}$ distribution.

Lemma 2.3.1 (Drowning [AIK11]). *Let $B_0 \in \mathbb{N}$ and $\beta \in \mathbb{Z}$ and let $e_0 \in [-B_0, B_0]$. Let $e_1 \leftarrow\!\$\ \Phi_{\mathbb{Z},\beta}$. If $B_0/\beta = \mathsf{negl}(\lambda)$ then $e_1 \approx_{\mathsf{negl}(\lambda)} e_0 + e_1$.*

2.4 Oblivious Degree Test for Rational Functions

Suppose we have a rational function $f(x) = P(x)/Q(x)$ where $P(x)$ and $Q(x)$ are two polynomials with the same degree. In this section, we present a protocol that allows several parties to check if $\deg P(x) = \deg Q(x) \leq t$ for some threshold $t \in Z$. To this end, and inspired by the works of [NW06, KMWF07], we present a multi-party protocol to obliviously solve a linear system $\mathbf{Mx} = \mathbf{y}$ over a finite field $\mathbb{F}$ with communication complexity $O(t^2 k\lambda N)$, where $\mathbf{M} \in \mathbb{F}^{t\times t}$, $\log|\mathbb{F}| = k$ and N is the number of parties involved in the protocol.

2.4.1 Oblivious Linear Algebra

In this section, we state the Secure Linear Algebra protocols that we need to build our degree test protocol. For the sake of briefness, the protocols are presented in Appendix A.1. These protocols all have the following form: There is a public key of a TPKE that encrypts a matrix $\mathbf{M}$ and every party involved in the protocol has a share of the secret key.

Note that if we let parties P_i input their encrypted matrix $\mathsf{Enc}(\mathbf{M})$, then the ideal functionality $\mathcal{F}$ has to know the secret key (by receiving secret key shares from all parties), otherwise $\mathcal{F}$ cannot compute the corresponding function correctly. However, this will cause an unexpected problem in security proof as mentioned in our introduction and [BMRR21]: The environment $\mathcal{Z}$ will learn the secret key as well since it can choose inputs for all parties. We fix this by relying on a global UC framework where exists a shared functionality $\bar{\mathcal{G}}$ in charge of distributing key pairs ($\mathcal{F}_{\mathsf{Gen}}$ from Section 1.5).

Oblivious matrix multiplication

We begin by presenting the ideal functionality for a multi-party protocol to jointly compute the product of two matrices, under a TPKE. The protocol is presented in Appendix A.1.1.

Ideal functionality. The ideal functionality for oblivious matrix multiplication is presented below.

$\mathcal{F}_{\mathsf{OMM}}$ functionality

Parameters: $\mathsf{sid}, N, q, t \in \mathbb{N}$ and $\mathbb{F}$, where $\mathbb{F}$ is a field of order q, known to the N parties involved in the protocol.

Global Setup: pk public-key of a threshold PKE scheme and sk_i distributed to each party P_i via $\mathcal{F}_{\mathsf{Gen}}$.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_1, \mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l), \mathsf{Enc}(\mathsf{pk}, \mathbf{M}_r))$ from party P_1 (where $\mathbf{M}_l, \mathbf{M}_r \in \mathbb{F}^{t\times t}$), $\mathcal{F}_{\mathsf{OMM}}$ outputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l \cdot \mathbf{M}_r)$ to P_1 and $(\mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l), \mathsf{Enc}(\mathsf{pk}, \mathbf{M}_r), \mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l \cdot \mathbf{M}_r))$ to all other parties P_i, for $i = 2, \ldots, N$.

Securely Compute the Rank of a Matrix

We present the ideal functionality to obliviously compute the rank of an encrypted matrix. The protocol is presented in Appendix A.1.2.

Ideal Functionality. The ideal functionality of oblivious rank computation is defined below.

$\mathcal{F}_{\mathsf{ORank}}$ functionality

Parameters: $\mathsf{sid}, N, q, t \in \mathbb{N}$ and $\mathbb{F}$, where $\mathbb{F}$ is a field of order q, known to the N parties involved in the protocol.

Global Setup: pk public-key of a threshold PKE scheme and sk_i distributed to each party P_i via $\mathcal{F}_{\mathsf{Gen}}$.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_1, \mathsf{Enc}(\mathsf{pk}, \mathbf{M}))$ from party P_1 (where $\mathbf{M} \in \mathbb{F}^{t \times t}$), $\mathcal{F}_{\mathsf{ORank}}$ outputs $\mathsf{Enc}(\mathsf{pk}, \mathrm{rank}(\mathbf{M}))$ to P_1 and $(\mathsf{Enc}(\mathsf{pk}, \mathbf{M}), \mathsf{Enc}(\mathsf{pk}, \mathrm{rank}(\mathbf{M}))$ to all other parties P_i, for $i = 2, \ldots, N$.

Oblivious Linear System Solver

We now show how N parties can securely solve a linear system using the multiplication protocol above. We follow the ideas from [KMWF07] to reduce the problem to minimal polynomials, and the only difference is we focus on multiparty settings.

The protocol is presented in Appendix A.1.5. Informally, we evaluate an arithmetic circuit following the ideas of [CDN01], and for the unary representation, a binary-conversion protocol [ST06] is required. All of the above protocols can be based on the Paillier cryptosystem.

Ideal Functionality. We give an ideal functionality of oblivious linear system solver for multiparty as follows.

$\mathcal{F}_{\mathsf{OLS}}$ functionality

Parameters: $\mathsf{sid}, N, q, t \in \mathbb{N}$ and $\mathbb{F}$, where $\mathbb{F}$ is a field of order q , known to the N parties involved in the protocol. pk public-key of a threshold PKE scheme.

Global Setup: pk public-key of a threshold PKE scheme and sk_i distributed to each party P_i via $\mathcal{F}_{\mathsf{Gen}}$.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_1, \mathsf{Enc}(\mathsf{pk}, \mathbf{M}), \mathsf{Enc}(\mathsf{pk}, \mathbf{y}))$ from party P_1 (assuming there is a solution $\mathbf{x}$ for $\mathbf{Mx} = \mathbf{y}$), $\mathcal{F}_{\mathsf{OLS}}$ outputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{x})$ such that $\mathbf{Mx} = \mathbf{y}$.

2.4.2 Oblivious Degree Test

We now present the main protocol of this section and the one that will be used in the construction of threshold PSI. Given a rational function $P(x)/Q(x)$ (for two polynomials $P(x)$ and $Q(x)$ with the same degree) and two support sets V_1, V_2, the protocol allows us to test if the degree of the polynomials is less than some threshold t. Of course, we can do this using generic approaches like garbled circuits. However, we are interested in solutions with communication complexity depending on t (even when the degree of $P(x)$ or $Q(x)$ is much larger than t).

Ideal functionality. The ideal functionality for the degree test of rational functions is presented below.

$\mathcal{F}_{\mathsf{SDT}}$ functionality

Parameters: $\mathsf{sid}, N, q, n, t \in \mathbb{N}, \mathbb{F}$ is a field of order q and t is a pre-defined threshold, known to the N parties involved in the protocol. pk public-key of a threshold PKE scheme. $\alpha_1, \dots, \alpha_{4t+2} \leftarrow\$ \mathbb{F}$ known to the N parties.

Global Setup: pk public-key of a threshold PKE scheme and sk_i distributed to each party P_i via $\mathcal{F}_{\mathsf{Gen}}$.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_1, \mathsf{Enc}(\mathsf{pk}, f_1), \dots, \mathsf{Enc}(\mathsf{pk}, f_{4t+2}))$ from party P_1 (where $f_i = P_1(\alpha_i)/P_2(\alpha_i)$, and P_1, P_2 are two co-prime polynomials with same degree t' (additionally, P_2 is monic), $\mathcal{F}_{\mathsf{SDT}}$ outputs 0 if $t' \leq t$; otherwise it outputs 1.

Protocol. We present the Protocol 1 for secure degree test which we denote by secDT. The main idea of the protocol is to interpolate the rational function on two different support sets and check if the result is the same in both experiments.

Recall that interpolating a rational function boils down to solving a linear equation. We can thus use the secure linear algebra tools developed to allow the parties to securely solve a linear equation.

Also recall that two rational functions $C_v^{(1)}/C_v^{(2)} = C_w^{(1)}/C_w^{(2)}$ are equivalent if $C_v^{(1)}C_w^{(2)} - C_w^{(1)}C_v^{(2)} = 0$. Thus, in the end, parties just need to securely check if $C_v^{(1)}C_w^{(2)} - C_w^{(1)}C_v^{(2)}$ is equal to 0.

Comments. Suppose that, for an interpolation point α_i, the rational function $f(x) = P(x)/Q(x)$ is well-defined but $Q(\alpha_i) = P(\alpha_i) = 0$ such that we cannot compute $f(\alpha_i)$ by division. In this case [8], the parties evaluate $\tilde{P}(x) = P(x)/(x - \alpha_i)$ and $\tilde{Q}(x) = Q(x)/(x - \alpha_i)$ on α_i and set $f(\alpha_i) = \tilde{P}(\alpha_i)/\tilde{Q}(\alpha_i)$. These points are called *tagged values* and this strategy is used in [MTZ03]. In more details, instead of using $\mathsf{Enc}(\mathsf{pk}, f_i)$ for α_i, we will use a tagged pair $\left(\mathsf{Enc}\left(\mathsf{pk}, s_i^{(1)}\right), \mathsf{Enc}\left(\mathsf{pk}, s_i^{(2)}\right)\right)$

[7]Note that this is the linear system that we need to solve in order to perform rational interpolation [MTZ03].

[8]In the case that only $Q(\alpha_i) = 0$, use a different tagged pair $(\mathsf{Enc}(\mathsf{pk}, s_i^{(1)}), \mathsf{Enc}(\mathsf{pk}, 0))$, and this can be noticed by the party who owns polynomial $Q(x)$. In our PSI setting, it is party P_1.

Algorithm 1 Secure Degree Test secDT

```
Require: Each party has a secret key share $\mathsf{sk}_i$ for a public key $\mathsf{pk}$ of a $\mathsf{TPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The parties have access to the ideal functionalities $\mathcal{F}_{\mathsf{ORank}}$, $\mathcal{F}_{\mathsf{OLS}}$, $\mathcal{F}_{\mathsf{OMM}}$ and $\mathcal{F}_{\mathsf{DecZero}}$. The values $\{\alpha_1, \ldots, \alpha_{4t+2}\} \leftarrow\!\$\ \mathbb{F}^{4t+2}$ are public, from which also sampling a random point $\alpha' \leftarrow\!\$\ \{\alpha_1, \ldots, \alpha_{4t+2}\}$.
Ensure: Party $\mathsf{P}_1$ inputs $\{(\alpha_1, \mathsf{Enc}(\mathsf{pk}, f_1)), \ldots, (\alpha_{4t+2}, \mathsf{Enc}(\mathsf{pk}, f_{4t+2}))\}$, where $f_i = \frac{P_1(\alpha_i)}{P_2(\alpha_i)}$, where $P_1(x), P_2(x)$ are two polynomials with degree $\deg(P_1) = \deg(P_2) = t' = \mathsf{poly}(\log|\mathbb{F}|)$ and such that $P_2(\alpha_i) \neq 0$ for all $i \in [2t]$.
1: $\mathsf{P}_1$ sets $\{(\alpha_j, \mathsf{Enc}(\mathsf{pk}, f_j))\}_{j\in[2t+1]} = \{(v_j, \mathsf{Enc}(\mathsf{pk}, f_{v,j}))\}_{j\in[2t+1]}$, and $\{(\alpha_j, \mathsf{Enc}(\mathsf{pk}, f_j))\}_{j\in\{2t+2,\ldots,4t+2\}} = \{(w_j, \mathsf{Enc}(f_{w,j}))\}_{j\in[2t+1]}$. It homomorphically generates an encrypted linear system consisting of
   $$\mathsf{Enc}(\mathsf{pk}, \mathbf{M}_r) = \mathsf{Enc}\left(\mathsf{pk}, \begin{bmatrix} r_1^t & \ldots & 1 & -f_{r,1}\cdot r_1^{t-1} & \ldots & -f_{r,1} \\ \vdots & & \vdots & \vdots & & \vdots \\ r_{2t+1}^t & \ldots & 1 & -f_{r,2t+1}\cdot r_{2t+1}^{t-1} & \ldots & -f_{r,2t+1} \end{bmatrix}\right)$$
   and
   $$\mathsf{Enc}(\mathsf{pk}, \mathbf{y}_r) = \mathsf{Enc}\left(\mathsf{pk}, \begin{bmatrix} f_{r,1}\cdot r_1^t \\ \vdots \\ f_{r,2t+1}\cdot r_{2t+1}^t \end{bmatrix}\right)$$
   for $r = \{v, w\}$.[7] Here $\mathbf{M}_r$ is a square matrix with dimension $2t+1$ and $\mathbf{y}_r$ a $2t+1$-sized vector.
2: All parties jointly compute $\mathsf{Enc}(\mathsf{pk}, \mathrm{rank}(\mathbf{M}_r) - \mathrm{rank}([\mathbf{M}_r||\mathbf{y}])$ for $r \in \{v, w\}$ through two invocations of $\mathcal{F}_{\mathsf{ORank}}$ and mutually decrypt the ciphertext via $\mathcal{F}_{\mathsf{DecZero}}$. If the result is different from 0, they abort the protocol.
3: All parties mutually solve the two linear systems above using $\mathcal{F}_{\mathsf{OLS}}$ such that each party gets $\mathsf{Enc}\left(\mathsf{pk}, \left(\mathbf{c}_v^{(1)}||\mathbf{c}_v^{(2)}\right)\right)$ and $\mathsf{Enc}\left(\mathsf{pk}, \left(\mathbf{c}_w^{(1)}||\mathbf{c}_w^{(2)}\right)\right)$, where $\mathbf{M}_r \begin{bmatrix} \mathbf{c}_r^{(1)} \\ \mathbf{c}_r^{(2)} \end{bmatrix} = \mathbf{y}_r$, for $r \in \{v, w\}$. Besides, $\mathbf{c}_r^{(1)}$ and $\mathbf{c}_r^{(2)}$ are $t+1$- and $t$-sized vectors, respectively.
4: All parties compute the polynomials $C_r^{(1)}(x) = \sum_{j=0}^{t} \mathbf{c}_{r,j}^{(1)} x^{t-j}$, and $C_r^{(2)}(x) = x^t + \sum_{j=1}^{t} \mathbf{c}_{r,j-1}^{(2)} x^{t-j}$, for $r \in \{v, w\}$, then compute
   $$\mathsf{Enc}(\mathsf{pk}, z) = \mathsf{Enc}(\mathsf{pk}, C_v^{(1)}(x)\cdot C_w^{(2)}(x) - C_w^{(1)}(x)\cdot C_v^{(2)}(x))$$
   by invoking $\mathcal{F}_{\mathsf{OMM}}$.
   Here $C_r^{(b)}(x)$ are evaluated on a random selected point $\alpha' \leftarrow\!\$\ \{\alpha_1, \ldots, \alpha_{4t+2}\}$.
5: All parties jointly use $\mathcal{F}_{\mathsf{DecZero}}$ to check if $z = 0$. If it is, output 1. Otherwise, output 0.
```

where $s_i^{(1)} = \frac{P_1(\alpha_i)}{x-\alpha_i}$ and $s_i^{(2)} = \frac{P_2(\alpha_i)}{x-\alpha_i}$. Correspondingly, replace each row of $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}_r)$ and $\mathsf{Enc}(\mathsf{pk}, \mathbf{y}_r)$ with

$$\mathsf{Enc}\left(\mathsf{pk}, \left[s_i^{(2)} r_i^t \quad \dots \quad s_i^{(2)} \quad -s_i^{(1)} r_i^{t-1} \quad \dots \quad -s_i^{(1)}\right]\right)$$

and $\mathsf{Enc}\left(\mathsf{pk}, \left[s_i^{(1)} r_i^t\right]\right)$, respectively.

Also, note that the protocol easily generalizes to rational functions $f(x) = P(x)/Q(x)$ with $\deg P \neq \deg Q$ (which is actually what we use in the following sections). We present the version where $\deg P = \deg Q$ for simplicity. In fact, the case where $\deg P \neq \deg Q$ can be reduced to the presented case by multiplying the least degree polynomial by a uniformly chosen $R(x)$ of degree $\max\{\deg P(x) - \deg Q(X), \deg Q(x) - \deg P(x)\}$.

Moreover, if $t' > t$, the linear system for rational interpolation might be unsolvable. In this case, there is no solution which means we cannot interpolate an appropriate rational function on certain support set. Therefore, the parties just return 0.

Analysis We analyze correctness, security and communication complexity of the protocol. We begin the analysis with the following auxiliary lemma.

Lemma 2.4.1. *Let $\mathbb{F}$ be a field with $|\mathbb{F}| = \omega(2^{\log \lambda})$. Let $V = \{(v_i, f(v_i)) | \forall i \in [1, 2t+1]\}$ and $W = \{(w_i, f(w_i)) | \forall i \in [1, 2t+1]\}$ be two support sets each of them with $2t+1$ elements over a field $\mathbb{F}$, with $w_i \leftarrow\$ \mathbb{F}$, and $f(x) := \frac{P(x)}{Q(x)}$ is some unknown reduced rational function (i.e., $P(x), Q(x)$ are co-prime), where $\deg(P) = \deg(Q) = t'$ and $t < t'$ where $t, t' \in \mathsf{poly}(\lambda)$. We also require $Q(x)$ to be monic (to fit in our application). Additionally, assume that $Q(v_i) \neq 0$ and $Q(w_i) \neq 0$ for every $i \in [2t+1]$.*

If we recover two rational function $f_V(x), f_W(x)$ by interpolation on V, W, respectively, then

$$\Pr[f_V(x) = f_W(x)] \leq \mathsf{negl}(\lambda)$$

over the choice of v_i, w_i.

Proof. Let $f_V(x) = A(x)/B(x)$ the rational function recovered by rational interpolation over the support set V. and let $f(x) = P(x)/Q(x)$ be the rational function interpolated over any $2t'+1$ interpolation points. We have that $f_V(v_i) = f(v_i)$ for all $i \in [2t+1]$ and hence

$$\frac{A(v_i)}{B(v_i)} = \frac{P(v_i)}{Q(v_i)} \Leftrightarrow A(v_i)Q(v_i) = P(v_i)B(v_i).$$

Since $\gcd(P(x), Q(x)) = 1$, then the polynomial $\tilde{P}(x) = A(x)Q(x) - P(x)B(x)$ is different from the null polynomial (as $\deg(P) = t' > t = \deg(A)$). Moreover, v_i is a root of $\tilde{P}(x)$, for all $i \in [2t+1]$, and $\deg \tilde{P}(x) \leq t + t'$ (which means that $\tilde{P}(x)$ has at most $t + t'$ roots).

Analogously, let $f_W = C(x)/D(x)$ be the rational function resulting from interpolating over the support set W and let $\tilde{Q}(x) = C(x)Q(x) - D(x)P(x)$. We have that $\tilde{Q}(w_i) = 0$ for all $i \in [2t+1]$. Hence, if $f_V(x) = f_W(x)$, then we have that the points w_i are also roots of $\tilde{P}(x)$.

But, since the points w_i are chosen uniformly at random from $\mathbb{F}$ (which is of exponential size when compared to t, t'), then there is a negligible probability that all w_i's are roots of $\tilde{P}(x)$.

Concretely,

$$\begin{aligned}\Pr\left[f_V = f_W\right] &\leq \Pr\left[\tilde{P}(w_i) = 0 \forall i[2t+1]\right] \\ &= \prod_{i}^{2t+1} \Pr\left[\tilde{P}(w_i) = 0\right] \leq \left(\frac{\deg \tilde{P}}{|\mathbb{F}|}\right)^{2t+1}\end{aligned}$$

which is negligible for $|\mathbb{F}| \in \omega(2^{\log \lambda})$. □

Theorem 2.4.2 (Correctness). *The protocol* secDT *is correct.*

Proof. The protocol interpolates two polynomials from two different support sets. Then, it checks if the two interpolated polynomials are the same by computing

$$C_v^{(1)}(x) \cdot C_w^{(2)}(x) - C_w^{(1)}(x) \cdot C_v^{(2)}(x))$$

which should be equal to 0 if $C_v^{(1)}(x)/C_v^{(2)}(x) = C_w^{(1)}(x)/C_w^{(2)}(x)$.

If $t' \leq t$, then by Lemma 1.2.6, there is a unique rational function that can be recovered thus the final output of the algorithm should be 1. On the other hand, if $t' > t$, the linear system can be either unsolvable or solvable but yield two different solutions with overwhelming probability by Lemma 2.4.1. In this case, the protocol outputs 0. □

Theorem 2.4.3. *The protocol* secDT *EUC-securely realizes* $\mathcal{F}_{\mathsf{SDT}}$ *with shared ideal functionality* $\mathcal{F}_{\mathsf{Gen}}$ *in the* $(\mathcal{F}_{\mathsf{ORank}}, \mathcal{F}_{\mathsf{OMM}}, \mathcal{F}_{\mathsf{OLS}}, \mathcal{F}_{\mathsf{DecZero}})$*-hybrid model against semi-honest adversaries corrupting at most* $N-1$ *parties, given that* TPKE *is IND-CPA.*

Proof (Sketch). The simulator sends the corrupted parties' input to the ideal functionality and obtains the output (either 0 or 1). Then, it simulates the ideal functionalities $(\mathcal{F}_{\mathsf{ORank}}, \mathcal{F}_{\mathsf{OMM}}, \mathcal{F}_{\mathsf{OLS}}, \mathcal{F}_{\mathsf{DecZero}})$ so that the output in the real-world execution is the same as in the ideal-world execution. In particular, the simulator is able to recover the secret key shares via $\mathcal{F}_{\mathsf{ORank}}, \mathcal{F}_{\mathsf{OMM}}, \mathcal{F}_{\mathsf{OLS}}$ and, thus, simulate $\mathcal{F}_{\mathsf{DecZero}}$ in the right way.

Indistinguishability of executions holds given that TPKE is IND-CPA. □

Communication complexity. When we instantiate $\mathcal{F}_{\mathsf{OLS}}$ with the protocol from the previous section, the communication complexity of secDT is $\mathcal{O}(Nt^2)$.

2.5 Multi-Party Threshold Private Set Intersection

We present our protocol for Threshold PSI in the multi-party setting. Our protocol to privately compute the intersection can be seen as a generalization of Ghosh and Simkin protocol [GS19a] where we replace the OLE with a TPKE (which fits nicer in a multi-party setting). The main difference between our protocol and theirs is in the cardinality test protocol used.

We begin by presenting the protocol to securely compute cardinality testing between N sets. Then, we plug everything together in a PSI protocol.

2.5.1 Secure Cardinality Testing

Ideal functionality. The ideal functionality for Secure Cardinality Testing receives the sets from all the parties and outputs 1 if and only if the intersection between these sets is larger than some threshold. Else, no information is disclosed. The ideal functionality for multi-party cardinality testing is given as follows.

$\mathcal{F}_{\mathsf{MPCT}}$ functionality

Parameters: $\mathsf{sid}, N, n, t \in \mathbb{N}$ known to both parties.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_i, S_i)$ from party P_i, $\mathcal{F}_{\mathsf{MPCT}}$ stores S_i and ignores future messages from P_i with the same sid;
- Once $\mathcal{F}_{\mathsf{MPCT}}$ has stored all inputs S_i, for $i \in [N]$, it does the following: If $|S_\cap| \geq n - t$, $\mathcal{F}_{\mathsf{MPCT}}$ outputs 1 to all parties, where $|S_\cap| = \cap_{i=1}^{N} S_i$. Else, it returns 0.

Protocol. We introduce our multiparty Protocol 2 (based on degree test protocol). In the following, $\mathcal{F}_{\mathsf{Gen}}$ be the ideal functionality defined in Section 1.5 and $\mathcal{F}_{\mathsf{SDT}}$ be the functionality defined in Section 2.4.2.

Analysis. We now proceed to the analysis of the protocol described above. Note that $\mathcal{F}_{\mathsf{SDT}}$ has shared functionality $\mathcal{F}_{\mathsf{Gen}}$.

Lemma 2.5.1. *Given n characteristic polynomials with same degree from $\mathbb{F}[x]$, denoted as $P_1(x), \ldots, P_n(x)$, we argue that, for any j, $P'(x) = \sum_{i=1}^{n} r_i \cdot P_i(x)$ and $P_j(x)$ are relatively prime with probability $1 - \mathsf{negl}(\log |\mathbb{F}|)$ if $P_1(x), \ldots, P_n(x)$ are mutually relatively prime, where $r_i \leftarrow\$ \mathbb{F}$ is a uniformly random element.*

Proof. Supposing there is a common divisor of two polynomials $P'(x)$ and $P_j(x)$, since $P_j(x)$ is a characteristic polynomial, we denote $(x - s)$ the common divisor. Therefore, we have $P'(s) = 0$ which can be represented as $\sum_{i=1}^{n} r_i \cdot P_i(s) = 0$. However, from the mutually relative primality of

Algorithm 2 Private Cardinality Test for Multi-party MPCT

```
Require: Values $\alpha_1, \dots, \alpha_{4t+2} \leftarrow\$ \mathbb{F}$, threshold $t \in \mathbb{N}$ and $N$ parties. Functionalities $\mathcal{F}_{\mathsf{Gen}}$
    and $\mathcal{F}_{\mathsf{SDT}}$, and a IND-CPA TPKE $\mathsf{TPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
Ensure: Each party $P^i$ inputs a set $S_i = \{a_i^{(1)}, \dots, a_i^{(n)}\} \in \mathbb{F}^n$.
 1: Each party $\mathsf{P}_i$ sends request $(\mathsf{sid}, \mathsf{request}_i)$ to $\mathcal{F}_{\mathsf{Gen}}$ and receives a secret key share $\mathsf{sk}_i$ and
    a public key $\mathsf{pk}$, which is known to every party involved in the protocol.
 2: Each party $\mathsf{P}_i$ encodes its set as a polynomial $P_i(x) = \prod_{j=1}^{n}(x - a_i^{(j)})$ and evaluates it on
    $4t + 2$ points. That is, it computes $P_i(\alpha_1), \dots, P_i(\alpha_{4t+2})$. It encrypts the points, that
    is, $c_i^{(j)} \leftarrow \mathsf{Enc}(\mathsf{pk}, r_i \cdot P_i(\alpha_j))$ for a uniformly chosen $r_i \leftarrow\$ \mathbb{F}$. Finally, it broadcasts
    $\{c_i^{(j)}\}_{j\in[4t+2]}$.
 3: Party $\mathsf{P}_1$ computes $d^{(j)} = (\sum_{i=1}^{N} c_i^{(j)})/P_1(\alpha_j)$ for each $j \in [4t+2]$. Then, sends
    $\{\alpha_i, d^{(j)}\}_j$ for every $j$, and $\mathsf{sk}_1$ to the ideal functionality $\mathcal{F}_{\mathsf{SDT}}$. Each party $\mathsf{P}_i$, for $i =$
    $2, \dots, N$, send $\mathsf{sk}_i$ to $\mathcal{F}_{\mathsf{SDT}}$ to check if the degree of the numerator (and the denomina-
    tor) is at most $t$.
 4: Upon receiving $b \in \{0,1\}$ from the ideal functionality $\mathcal{F}_{\mathsf{SDT}}$, every party outputs $b$.
```

$P_1(x), \dots, P_n(x)$, we know that $P_i(s)$ cannot be zero simultaneously which means there exists at least one i^* to make $P_{i^*}(s) \neq 0$. Moreover, r_i are all sampled uniformly from $\mathbb{F}$, the weighted sum of r_i will not be zero with all but negligible probability. This is a contradiction. Therefore, $P'(x)$ and $P_j(x)$ will share a common divisor only with negligible probability. □

Theorem 2.5.2 (Correctness). *The protocol* MPCT *described above is correct.*

Proof. Note that the encryption $d^{(j)}$ computed by party P_1 are equal to

$$d^{(j)} = \mathsf{Enc}\left(\mathsf{pk}, \left(\sum_{i=1}^{N} r_i \cdot P_i(\alpha_j)\right) / P_1(\alpha_j)\right).$$

Also, observe that

$$\begin{aligned}\frac{\sum_{i=1}^{N} r_i \cdot P_i(\alpha_j)}{P_1(\alpha_j)} &= \frac{P_{\cap_i S_i}(\alpha_j) \cdot \sum_i^N r_i \cdot P_{S_i \setminus (\cap_{k\neq i} S_k)}(\alpha_j)}{P_{\cap_i S_i}(\alpha_j) \cdot P_{S_1 \setminus (\cap_{k\neq 1} S_k)}} \\ &= \frac{\sum_i^N r_i \cdot P_{S_i \setminus (\cap_{k\neq i} S_k)}(\alpha_j)}{P_{S_1 \setminus (\cap_{k\neq 1} S_k)}(\alpha_j)},\end{aligned}$$

in this way, we make the numerator and denominator relatively prime except with negligible probability by Lemma 2.5.1.

Observe that $\deg \sum_i^N r_i \cdot P_{S_i \setminus (\cap_{k \neq i} S_k)}(x) \leq t$ and $\deg P_{S_1 \setminus (\cap_{k \neq 1} S_k)}(x) \leq t$ if and only if $S_\cap \geq n - t$. Hence, by the correctness of $\mathcal{F}_{\mathsf{SDT}}$, the protocol outputs 1 if $S_\cap \geq n - t$, and 0 otherwise. $\square$

Theorem 2.5.3. *The protocol* MPCT *securely realizes functionality* $\mathcal{F}_{\mathsf{MPCT}}$ *in the* $(\mathcal{F}_{\mathsf{Gen}}, \mathcal{F}_{\mathsf{SDT}})$*-hybrid model against any semi-honest adversaries corrupting up to* $N - 1$ *parties, given that* TPKE *is IND-CPA.*

Proof. Assume that the adversary is corrupting $N - k$ parties in the protocol, for $k = 1, \ldots, N - 1$. The simulator creates the secret keys and the public key of a threshold PKE in the setup phase while simulating $\mathcal{F}_{\mathsf{Gen}}$ and distributes the secret keys between every party. The simulator Sim takes the inputs (which are sets of size n, say $S_{i_1}, \ldots, S_{i_{N-k}}$) of the corrupted parties and send them to the ideal functionality $\mathcal{F}_{\mathsf{MPCT}}$. It receives the output b from the ideal functionality. If $b = 0$, the simulator chooses k uniformly chosen sets such that $| \cap_{i=1}^N S_i| < n - t$ and proceed the simulation as the honest parties would do. If $b = 1$, , the simulator chooses k uniformly chosen random sets such that $|\cap_{i=1}^N S_i| \geq n - t$ and proceed the simulation as the honest parties would do. Note that it can simulate the ideal functionality $\mathcal{F}_{\mathsf{SDT}}$ since it knows all the secret keys of the threshold PKE.

Indistinguishability of executions follows immediately from the IND-CPA property of the underlying threshold PKE scheme. $\square$

Communication Complexity. When we instantiate the $\mathcal{F}_{\mathsf{SDT}}$ with the protocol from the previous section, each party broadcasts $\tilde{\mathcal{O}}(t^2)$. Hence, the total communication complexity is $\tilde{\mathcal{O}}(Nt^2)$, assuming a broadcast channel.

2.5.2 Multi-party Threshold Private Set Intersection Protocol

In this section, we extend Ghosh and Simkin protocol [GS19a] to the multi-party setting using TPKE. We make use of the cardinality testing designed above to get the Protocol 3.

Analysis. We now proceed to the analysis of the protocol described above. We start by analyzing the correctness of the protocol and then its security.

Theorem 2.5.4 (Correctness). *The protocol* MTPSI *is correct.*

Proof. Assume that $|S_1 \setminus \left(\cap_{i=2}^N S_i\right) | \leq t$ (note that this condition is guaranteed after resorting to the functionality $\mathcal{F}_{\mathsf{MPCT}}$ in the first step of the protocol). After the execution of the protocol, party P_1

Algorithm 3 Multi-Party Threshold PSI MTPSI

```
Require: Given public parameters as follows: Values $\alpha_1, \dots, \alpha_{3t+1} \leftarrow\$ \mathbb{F}$, threshold $t \in \mathbb{N}$ and $N$ parties. Functionalities $\mathcal{F}_{\mathsf{Gen}}$ and $\mathcal{F}_{\mathsf{MPCT}}$, and a threshold additively PKE $\mathsf{TPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
Ensure: Each party $\mathsf{P}_i$ inputs a set $S_i = \{a_i^{(1)}, \dots, a_i^{(n)}\} \in \mathbb{F}^n$.
 1: Each party $\mathsf{P}_i$ sends its set $S_i$ to $\mathcal{F}_{\mathsf{MPCT}}$. If the functionality $\mathcal{F}_{\mathsf{MPCT}}$ outputs $0$, then every party $\mathsf{P}_i$ outputs $\perp$ and terminates the protocol.
 2: Each party $\mathsf{P}_i$ sends request $(\mathsf{sid}, \mathsf{request}_i)$ to $\mathcal{F}_{\mathsf{Gen}}$ and receives a secret key share $\mathsf{sk}_i$ and a public key $\mathsf{pk}$, which is known to every party involved in the protocol.
 3: for all Party $\mathsf{P}_i$ do
 4:     It encodes its set as a polynomial $P_i(x) = \prod_{j=1}^{n}(x - a_i^{(j)})$ and evaluates it on $3t+1$ points. That is, it computes $P_i(\alpha_1), \dots, P_i(\alpha_{3t+1})$.
 5:     It samples $R_i(x) \leftarrow\$ \mathbb{F}[x]$ such that $\deg R_i(x) = t$.
 6:     It encrypts these points using $\mathsf{pk}$, that is, it computes $c_i^{(j)} = \mathsf{Enc}(\mathsf{pk}, R_i(\alpha_j) \cdot P_i(\alpha_j))$ for every $j \in [3t+1]$.
 7:     It broadcasts $\{c_i^{(j)}\}_{j \in [3t+1]}$.
 8: end for
 9: Party $\mathsf{P}_1$ adds the ciphertexts to get $d^{(j)} = \sum_i^N c_i^{(j)}$ for each $j \in [3t+1]$. It broadcasts $\{d^{(j)}\}_{j \in [3t+1]}$.
10: They mutually decrypt $\{d^{(j)}\}_{j \in [3t+1]}$ to learn $V^{(j)} \leftarrow \mathsf{Dec}(\mathsf{sk}, d_N^{(j)})$ for $j \in [3t+1]$.
11: $\mathsf{P}_1$ computes the points $\tilde{V}^{(j)} = V^{(j)}/P_1(\alpha_j)$ for $j \in [3t+1]$.
12: $\mathsf{P}_1$ interpolates a rational function using the pairs of points $(\alpha_j, \tilde{V}^{(j)})$.
13: $\mathsf{P}_1$ recovers the polynomial $P_{S_1 \setminus (\cap_i S_i)}(x)$ in the denominator.
14: $\mathsf{P}_1$ evaluates $P_{S_1 \setminus \cap_i S_i}(x)$ on every point of its set $\{a_1^{(1)}, \dots, a_1^{(n)}\}$ to compute $\cap_i S_i$. That is, whenever $P_{S_1 \setminus \cap_i S_i}(a_1^j) \neq 0$, then $a_1^j \in \cap_i S_i$.
15: It broadcasts the output $\cap_i S_i$.
```

obtains the points $V^{(j)} = \sum_i^N P_i(\alpha_j) \cdot R_i(\alpha_j)$. Then,

$$\begin{aligned}\tilde{V}^{(j)} = \frac{V^{(j)}}{P_1(\alpha_j)} &= \frac{\sum_i^N P_i(\alpha_j) \cdot R_i(\alpha_j)}{P_1(\alpha_j)} \\ &= \frac{P_{\cap_i S_i}(\alpha_j) \cdot \sum_i^N P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)}(\alpha_j) \cdot R_i(\alpha_j)}{P_{\cap_i S_i}(\alpha_j) \cdot P_{S_1 \setminus \left(\cap_{k \neq 1} S_k\right)}(\alpha_j)} \\ &= \frac{\sum_i^N P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)}(\alpha_j) \cdot R_i(\alpha_j)}{P_{S_1 \setminus \left(\cap_{k \neq 1} S_k\right)}(\alpha_j)}.\end{aligned}$$

Since P_1 has $3t+1$ evaluated points of the rational function above, then it can interpolate a rational function to recover the polynomial $P_{S_1 \setminus \left(\cap_{k \neq 1} S_k\right)}$. This is possible because of Lemma 1.2.5 and the fact that

$$\deg\left(\sum_i^N P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)}(\alpha_j) \cdot R_i(\alpha_j)\right) \leq 2t \quad \text{and} \quad \deg\left(P_{S_1 \setminus \left(\cap_{k \neq 1} S_k\right)}(\alpha_j)\right) \leq t.$$

Having computed the polynomial $P_{S_1 \setminus \left(\cap_{k \neq 1} S_k\right)}$, party P_1 can compute the intersection because the roots of this polynomial are exactly the elements in $S_1 \setminus \left(\cap_{k \neq 1} S_k\right)$. □

Theorem 2.5.5. *The protocol* MTPSI *securely realizes functionality* $\mathcal{F}_{\mathsf{MTPSI}}$ *in the* $(\mathcal{F}_{\mathsf{Gen}}, \mathcal{F}_{\mathsf{MPCT}})$-*hybrid model against any semi-honest adversary corrupting up to* $N-1$ *parties.*

Proof. Let $\mathcal{A}$ be an adversary corrupting up to k parties involved in the protocol, for any $k \in [N-1]$. Let $\mathsf{P}_{i_1}, \ldots, \mathsf{P}_{i_k}$ be the corrupted parties. The simulator Sim works as follows:

1. It sends the inputs of the corrupted parties, $S_{i_1}, \ldots, S_{i_k}$, to the ideal functionality $\mathcal{F}_{\mathsf{MTPSI}}$. Sim either receives $\perp$ or $\cap_i S_i$ from the ideal functionality $\mathcal{F}_{\mathsf{MTPSI}}$.

2. Sim waits for $\mathcal{A}$ to send the corrupted parties' inputs to the ideal functionality $\mathcal{F}_{\mathsf{MPCT}}$. If Sim has received $\perp$ from $\mathcal{F}_{\mathsf{MPCT}}$, then Sim leaks 0 to $\mathcal{A}$ (and $\mathcal{Z}$) and terminates the protocol. Else, Sim leaks 1 and continues.

3. Sim waits for $\mathcal{A}$ to send a request $(\mathsf{sid}, \mathsf{request}_{i_j})$ for each of the corrupted parties (that is, for $j \in [k]$) to $\mathcal{F}_{\mathsf{Gen}}$. Upon receiving such requests, Sim generates $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}(1^\lambda, N)$ and returns $(\mathsf{pk}, \mathsf{sk}_{i_j})$ for each of the requests.

4. For each party P_ℓ such that $\ell \neq i_j$ (where $j \in [k]$), Sim picks a random polynomial $U_\ell(x)$ of degree $n - |\cap_i S_i| + t$ and sends $\mathsf{Enc}(\mathsf{pk}, R_\ell(\alpha_j) \cdot P_{\cap_i S_i}(\alpha_j) \cdot U_\ell(\alpha_j))$, where $R_\ell(x)$ is chosen uniformly at random such that $\deg R_\ell(x) = t$. From now on, Sim simulates the dummy parties as in the protocol.

We now argue that both the simulation and the real-world scheme are indistinguishable from the point-of-view of any environment $\mathcal{Z}$. In the real-world scheme, party P_1 obtains the polynomial

$$V(x) = P_{\cap_i S_i}(x) \cdot \sum_{i}^{N} P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)}(x) \cdot R_i(x)$$

evaluated in $3t+1$ points. Assume that P_1 is corrupted by $\mathcal{A}$. Even in this case, there is an index ℓ for which $\mathcal{A}$ does not know the polynomial $R_\ell(x)$. More precisely, we have that

$$V(x) = P_{\cap_i S_i}(x) \cdot \left(\left(\sum_{i \neq \ell} P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)}(x) \cdot R_i(x) \right) + P_{S_\ell \setminus \left(\cap_{k \neq \ell} S_k\right)}(x) \cdot R_\ell(x) \right).$$

First, note that

$$\begin{aligned} \deg \left(\sum_{i \neq \ell} P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)}(x) \cdot R_i(x) \right) &= \deg P_{S_\ell \setminus \left(\cap_{k \neq \ell} S_k\right)}(x) \cdot R_\ell(x) \\ &= n - |\cap_i S_i| + t \leq 2t. \end{aligned}$$

Moreover, we have for any $i \in [N]$ that $\deg P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)} \leq t$, $\deg R_i(x) = t$ and $\gcd\left(P_{S_i \setminus \left(\cap_{k \neq i} S_k\right)}, P_{S_j \setminus \left(\cap_{k \neq j} S_k\right)}\right) = 1$ for any $j \neq i$. Hence, by Lemma 1.2.4, we can build a sequence of hybrids where we replace $V(x)$ by the polynomial $V'(x) = P_{\cap_i S_i}(x) \cdot U(x)$, where $\deg U(x) = n - |\cap_i S_i| + t$, as in the ideal-world execution. Indistinguishability of executions follows. $\square$

Communication complexity. When we instantiate the ideal functionality $\mathcal{F}_{\mathsf{MPCT}}$ with the protocol from the previous section the scheme has communication complexity $\tilde{\mathcal{O}}(Nt^2)$.

3 Laconic Private Set Intersection

In this chapter, we move on to discuss the second PSI-related problem in privacy-preserving computation, concentrating on communication bandwidth in an unbalanced setting. We provide a brief overview of this problem below.

Consider a server with a *large* set S of strings $\{x_1, x_2 \ldots, x_N\}$ that would like to publish a *small* hash h of its set S such that any client with a string y can send the server a *short* message allowing it to learn y if $y \in S$ and nothing otherwise. In this chapter, we study this problem of two-round private set intersection with low (asymptotically optimal) communication cost, or what we call *laconic* private set intersection (ℓPSI) and its extensions. This problem is inspired by the recent general frameworks for laconic cryptography [CDG⁺17, QWW18].

We start by showing the first feasibility result for realizing ℓPSI based on the CDH assumption, or LWE with polynomial noise-to-modulus ratio[1]. However, these feasibility results use expensive non-black-box cryptographic techniques leading to significant inefficiency. Next, with the goal of avoiding these inefficient techniques, we give a construction of ℓPSI schemes making only black-box use of cryptographic functions. Our construction is secure against semi-honest receivers, and malicious senders and reusable in the sense that the receiver's message can be reused across any number of executions of the protocol. The scheme is secure under the φ-hiding, decisional composite residuosity and subgroup decision assumptions.

At the end of this chapter, we show natural applications of ℓPSI to realize a semantically-secure encryption scheme that supports the detection of encrypted messages belonging to a set of "illegal" messages (e.g., an illegal video) circulating online. Over the past few years, significant effort has gone into realizing laconic cryptographic protocols. Nonetheless, this thesis provides the first black-box constructions of such protocols for a natural application setting.

[1] Refer to Chapter 1 for details about these hardness assumptions.

3.1 Overview

Laconic cryptography [CDG+17, QWW18, DGI+19, DGGM19] is an emerging paradigm which enables realizing cryptographic tasks with asymptotically-optimal communication in just two messages. In this setting, the receiver has a potentially large input, and the size of her protocol message only depends on the security parameter and not her input size. The second message, sent by the sender, may grow with the size of the sender's input but should be independent of the receiver's input size.

The pioneering work of [CDG+17] introduced the notion of laconic oblivious transfer (laconic OT), which allows a receiver with a large input $D \in \{0,1\}^n$ to send a short hash digest h of her input D. Next, a sender with an input $(i \in [n], m_0, m_1)$, sends a short message ots to the receiver, enabling the receiver to learn $m_{D[i]}$, and nothing more. We require (a) the sizes of h and ots be $\mathsf{poly}(\log(n), \lambda)$, where λ is the security parameter; (b) the sender's computation time be $\mathsf{poly}(\log(n), \lambda)$ and (c) and receiver's second-phase computation time be $\mathsf{poly}(\log(n), \lambda)$.

The notion of laconic OT, and the techniques built around it, have led to breakthrough results in the last few years, which, among others, include the first construction of identity-based encryption from CDH [DG17b, DG17a, BLSV18, DGHM18], and two-round MPC protocols from minimal assumptions [GS17, GS18, BL18].

Laconism beyond OT? Motivated by the developments enabled by laconic OT, it is natural to ask whether we can push the boundary further, realizing laconism for richer functionalities. Laconic OT by itself does not seem to be sufficient for this task (at least generically). Specifically, the general laconic OT+garbled circuit-based approach for a function $f(\cdot, \cdot)$ results in protocols in which the size of the sender's protocol message grows with the receiver's input size.

The work of Quach, Wee and Wichs [QWW18] shows how to realize laconic cryptography for general functionalities using LWE. However, two significant issues remain. Firstly, it is not clear whether we can achieve laconism from other assumptions, for functionalities beyond OT. As mentioned above, research in laconic OT has led to several breakthrough feasibility results, motivating the need for developing techniques that can be realized using wider assumptions and for richer functionalities. Secondly, existing constructions of laconic primitives are non-black-box, leading to inefficient constructions. Addressing the above shortcomings, our goals are twofold: (1) Feasibility: Can we realize laconic primitives beyond OT from assumptions other than LWE? and (2) Black-boxes: Can we make the constructions black-box?

Black-box techniques. We use the notion of "black-box" techniques in the sense that the construction should not use an explicit circuit-level description of cryptographic primitives. In this sense, we think of constructions which e.g., compute cryptographic primitives inside garbled circuits (as previous laconic OT constructions) or use general-purpose NIZK proofs (which express statements in terms of NP-complete languages) as "non-black-box" techniques.

Laconic PSI. We make the first progress toward the above two goals with respect to a non-trivial functionality: Laconic Private Set Intersection (ℓPSI) and its family where the private set intersection is recalled in Chapter 2.

Laconic PSI allows a receiver to send a short digest of its large data set, which in turn can be used by a sender to compute a PSI second-round message. We require that the total communication complexity as well as the sender's running time be independent of the receiver's input size.

3.1.1 Results

As our first result, we give a generic construction of laconic PSI from a primitive called anonymous hash encryption, which in turn can be realized from CDH/LWE [DG17b, DG17a, BLSV18]. Our construction builds on the Merkle-tree garbled circuit-based approach of [DG17b, DG17a, BLSV18, GHMR18, GHM$^+$19, GV20], showing how to use garbled circuits to perform binary search on a set of sorted values. Prior to our work there did not exist any construction of a laconic primitive from CDH beyond OT. We also obtain an LWE instantiation with polynomial modulus to noise ratio, improving the subexponential ratio of [QWW18].

The above construction is a non-black-box caused by the use of garbled circuits. As our second contribution, we achieve a black-box construction of laconic PSI from the φ-hiding assumption.

Both constructions above are only semi-honest secure, and can be made malicious (UC) secure by using Non-Interactive Zero Knowledge (NIZK).[2] However, the eventual protocol will be non-black-box. To enhance applicability, we show how to make our second construction secure against malicious senders, and semi-honest receivers in the CRS model, by additionally assuming decisional composite residuosity (DCR) and subgroup decision assumptions. We term this notion *reusable malicious* laconic PSI, meaning the receiver's message may be re-used.[3]

Applications. We show an application of laconic PSI in realizing a primitive that we dub self-detecting encryption. Self-detecting encryption acts like normal public-key encryption with a key difference in that it is possible to detect whether the underlying message of a given ciphertext belongs to a database of special (e.g., "illegal") messages. This can be determined just by knowing the database values, as opposed to the system's secret key. Such encryption systems provide a feature for detecting the presence of illegal content, without compromising the privacy of legal messages. There has only been a limited number of proposals for this task so far, and all of them use heavy tools (e.g., FHE) for this purpose (see [Gre19] for more details). We formally define this notion and show how to realize it using laconic PSI.

[2]Note that in the laconic setting, we cannot prove malicious security against a receiver since it is information-theoretically impossible to extract its input. Thus, since the NIZK will only be computed by the sender, the protocol will remain laconic.

[3]We use the word reusability only in conjunction with malicious security since in the semi-honest setting, reusability is satisfied by default.

In self-detecting encryption, an authority (e.g., a government entity or a delegated NGO) publishes a small hash value of a (possibly large) database of special messages such that a user can encrypt a message using the system's public key and the hash value.

If the message belongs to the database, then the authority can detect it; else, the message remains hidden from the authority. We require that the size of the hash and the encryption running time be independent of the database size.

We note that attribute-based encryption does not provide a solution to the above problem, because either the authority should reveal its database to a master-key generator, or it should be the master-key generator itself – both of which defeat our security purposes.

Additional new results: Labeled laconic PSI and malicious laconic OT (LOT). We extend our laconic PSI techniques to build a reusable *labelled* laconic PSI. Labelled PSI [JL10, CHLR18] is a flavour of PSI, where the sender holds a *label* ℓ_i associated with each set element x_i, and the receiver will learn the labels corresponding to the intersection elements. Labelled PSI has several practical applications (e.g., private web service queries [CHLR18]).

Moreover, we show how to use our techniques to realize the first construction of a reusable LOT secure against malicious senders and semi-honest receivers.

DV-NIZK range proofs for DJ ciphertexts. As a building block for our laconic PSI protocol, we propose a Designated-Verifier Non-Interactive Zero-Knowledge (DV-NIZK[4]) scheme for range proof with Damgård Jurik (DJ) ciphertexts, which may be of independent interest. Our DV-NIZK has statistical simulation soundness and computational zero-knowledge given that the subgroup decision (SD) assumption holds [BGN05, GOS06].

Such range proofs can also be constructed in the random oracle model (ROM) via the Fiat-Shamir transform (e.g., [DJ01, BBC$^+$18, BBB$^+$18, TBM$^+$20]), which might yield the best efficiency. As our LPSI construction is modular, this can be done independently of the remaining results in the paper. The goal of our DV-NIZK is to provide an efficient standard model construction which we see as a reasonable middle ground between feasibility from the weakest assumption (at the cost of unrealistic efficiency) and practical efficiency (at the cost of relying on strong heuristic assumptions such as the ROM).

3.1.2 Previous Work

Laconic PSI can be seen as a particular case of unbalanced PSI. Protocols for unbalanced PSI were presented in [ADT11, RA18, CLR17, CHLR18]. The protocol of [RA18] achieves linear communication complexity on the receiver's set size in the *pre-processing* model. The protocols of [CLR17, CHLR18] rely on somewhat homomorphic encryption (SWHE) and proceed in two rounds. However, the communication complexity scales with the size of the receiver's set (and logarithmic with the

[4]DV-NIZK *only* allows the designated prover to prove that it holds a witness for a certain NP statement to a verifier in just one message

size of the sender's set), in contrast with our protocol whose communication complexity scales with the sender's set size.

Comparison with [ADT11]. Ateniese et al. in [ADT11] proposed a semi-honest size-hiding PSI protocol[5] inspired by RSA accumulators that achieve communication complexity independent of the receiver's set size. However, we emphasize that their scheme does not fit the framework of laconic cryptography since it requires the sender to know the factorization of a CRS modulus N. Thus, either it requires pre-processing (giving a designated secret key to the sender), or it requires three rounds in the CRS model. In contrast, laconic cryptography requires (a) two rounds and (b) no pre-processing (i.e., neither party receives a secret key correlated with the CRS). Both (a) and (b) are crucially used in applications of laconic cryptography. Specifically, these restrictions prevent the use of [ADT11] in settings with multiple senders, an aspect that has been critical for laconic cryptography applications. Finally, we remark that the security of [ADT11] relies on random oracles, whereas we prove security in the standard model and achieve a substantially stronger security notion without resorting to heavy generic tools.

All of the above constructions are just secure against semi-honest adversaries, except for [CHLR18] which achieves security against a malicious receiver.

3.1.3 Open Problems

The main open question is to realize laconic cryptography for functionalities richer than PSI. A second question is to build laconic PSI in a black-box way from assumptions not involving φ-hiding (e.g., pairings alone).

In this chapter, we build DV-NIZK for proving the equality of plaintexts across different encryption schemes, namely between the DJ [DJ01] and the BGN [BGN05, GOS06] encryption schemes. This scheme opens the door to new applications since it allows us to extend the capabilities of GS/GOS proof systems [GOS06, GS08] to non-pairing-based primitives with additional properties (in our case to the DJ cryptosystem). We believe that these ideas will have applications beyond range proofs, e.g., one can think of further uses of structure-preserving cryptography, so we leave this as an open problem for future works.

3.2 Techniques

3.2.1 Semi-Honest PSI from CDH/LWE

Our protocol uses hash encryption and garbled circuits, building on [DG17b, BLSV18, GHMR18], while introducing new techniques. A hash-encryption scheme allows one to encrypt a message m to the output h of a hash function by specifying an index/bit (i, b) (denoted $\mathsf{HEnc}(h, m, (i, b))$), so that knowledge of a consistent pre-image value z allows for decryption ($\mathsf{Hash}(z) = h$ and $z_i = b$) while

[5] Such schemes were also studied in [IP07, LNO13, HW15].

having semantic security against inconsistent pre-image values (i.e., against z where $\mathsf{Hash}(z) = h$ but $z_i = \bar{b}$).[6]

In all discussion below we assume the sender's and receiver's elements are in $\{0,1\}^\lambda$ and that the output of Hash also has λ bits.

Receiver's set size is 2. We first assume the receiver has only two elements $S_\mathsf{R} = \{\mathsf{id}_1, \mathsf{id}_2\}$ and the sender has a single element id. The receiver sends $\mathsf{hr}_\mathsf{root} := \mathsf{Hash}(\mathsf{id}_1, \mathsf{id}_2)$. Consider a circuit $\mathsf{F}[\mathsf{id}]$, with id hardwired, which on input $(\mathsf{id}', \mathsf{id}'')$ outputs id if $\mathsf{id} \in \{\mathsf{id}', \mathsf{id}''\}$; else, $\perp$. The sender garbles $\mathsf{F}[\mathsf{id}]$ to get $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_{i,b}\})$[7] and sends $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \{\mathsf{ct}_{i,b}\})$, where $\mathsf{ct}_{i,b} := \mathsf{HEnc}(\mathsf{hr}_\mathsf{root}, \mathsf{lb}_{i,b}, (i,b)))$. The receiver who has the pre-image $z := (\mathsf{id}_1, \mathsf{id}_2)$ can retrieve only the labels lb_{i,z_i}, and the rest will be hidden. Thus, by garbled circuit security, the receiver will only learn the output of $\mathsf{F}[\mathsf{id}](\mathsf{id}_1, \mathsf{id}_2)$, as desired.

Moving beyond $|S_\mathsf{R}| = 2$. Suppose the receiver has four elements $S_\mathsf{R} = \{\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_3, \mathsf{id}_4\}$ in ascending order. The receiver Merkle-hashes all these values and sends $\mathsf{hr}_\mathsf{root}$, the root hash. Let h_1 and h_2 be the two hash values at level one (i.e., $h_1 = \mathsf{Hash}(\mathsf{id}_1, \mathsf{id}_2)$). If the sender knows the value of, say, h_1, he may hash-encrypt $\{\mathsf{lb}_{i,b}\}$ (defined in the previous paragraph) under h_1, so that the receiver can only open the labels that correspond to the bits of $z = (\mathsf{id}_1, \mathsf{id}_2)$, revealing the value of $\mathsf{F}[\mathsf{id}](\mathsf{id}_1, \mathsf{id}_2)$. However, h_1 is statistically hidden given $\mathsf{hr}_\mathsf{root}$. Thus, we use the idea of deferred evaluation [DG17b, CDG+17, DG17a, BLSV18], delegating the task of hash-encrypting $\{\mathsf{lb}_{i,b}\}$ to the receiver herself, via garbled circuits.

In essence, we want the receiver to be able to compute the hash encryption of $\{\mathsf{lb}_{i,b}\}$ wrt either h_1 or h_2 (depending on whether $\mathsf{id} \le \mathsf{id}_2$ or not), but not both; because obtaining both hash encryptions will allow the receiver to open both labels $\mathsf{lb}_{i,0}$ and $\mathsf{lb}_{i,1}$ for some indices i (because $(\mathsf{id}_1, \mathsf{id}_2) \neq (\mathsf{id}_3, \mathsf{id}_4)$), destroying garbled circuit security. Thus, the sender has to make sure that the receiver will be able to obtain only either of the above hash encryptions, the one whose sub-tree contains id. To enable this, we perform a binary search.

Performing binary search. We handle the above difficulty by performing *binary search* using ideas developed in the context of registration-based encryption [GHMR18]. The hash of each node is now computed as the hash of the concatenation of its left child's hash, right child's hash, and the largest identity under its left child. For example, the hash root is $\mathsf{hr}_\mathsf{root} = \mathsf{Hash}(h_1, h_2, \mathsf{id}_2)$, where h_1 and h_2 are the hash values of the two nodes in the first level, and in turn, $h_1 = \mathsf{Hash}(\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$. Now let id be the sender's element, and change $\mathsf{F}[\mathsf{id}]$ to be a circuit that on input $(\mathsf{id}', \mathsf{id}'', *)$ outputs id if $\mathsf{id} \in \{\mathsf{id}', \mathsf{id}''\}$, else $\perp$. Letting $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_{i,b}\})$ be the garbling of $\mathsf{F}[\mathsf{id}]$, consider a circuit $\mathsf{G}[\mathsf{id}, \{\mathsf{lb}_{i,b}\}]$ which on input (h, h', id') outputs a hash-encryption of $\mathsf{lb}_{i,b}$ either under h or under h', depending on whether $\mathsf{id} \le \mathsf{id}'$ or $\mathsf{id} > \mathsf{id}'$. Let $(\widetilde{\mathsf{C}'}, \{\mathsf{lb}'\}_{i,b})$ be the garbling of $\mathsf{G}[\mathsf{id}, \{\mathsf{lb}_{i,b}\}]$, let

[6] Enc also takes as input a public parameter p, which we ignore here.

[7] $\widetilde{\mathsf{C}}_0$ stands for the garbled circuit and $\{\mathsf{lb}_{i,b}\}_i$ are the corresponding labels of inputs.

$\{\mathsf{ct}_{i,b}\}$ be the hash encryption of $\{\mathsf{lb}'_{i,b}\}$ wrt $\mathsf{hr}_{\mathsf{root}}$, and return $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \widetilde{\mathsf{C}}', \{\mathsf{ct}_{i,b}\})$. Using the pre-image $z := (h_1, h_2, \mathsf{id}_2)$ of $\mathsf{hr}_{\mathsf{root}}$, the receiver can retrieve the labels $\{\mathsf{lb}'_{i,z[i]}\}$, allowing to compute $\mathsf{G}[\mathsf{id}, \{\mathsf{lb}_{i,b}\}](h_1, h_2, \mathsf{id}_2)$, which will produce a hash encryption $\{\mathsf{ct}'_{i,b}\}$ of $\{\mathsf{lb}_{i,b}\}$ under either h_1 or h_2, depending on whether $\mathsf{id} \le \mathsf{id}_2$, or not. For concreteness, suppose $\mathsf{id} \le \mathsf{id}_2$, meaning that $\{\mathsf{ct}'_{i,b}\}$ are formed under h_1, and so the pre-image $z' = (\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$ of h_1 will lead to $\{\mathsf{lb}_{i,z'_i}\}$, which along with $\widetilde{\mathsf{C}}_0$ will reveal the value of $\mathsf{F}[\mathsf{id}](\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$. Of course, the receiver *a priori* does not know whether $\{\mathsf{ct}'_{i,b}\}$ are encryptions under h_1 or h_2, so the receiver should try decrypting wrt both, and see which one succeeds.

Are we done? Unfortunately, when arguing about security, a subtle issue emerges. Suppose a hash-encryption ciphertext reveals its hash value (e.g., the hash is appended to the ciphertext). Then, the ciphertexts $\{\mathsf{ct}'_{i,b}\}$ will reveal whether they were encrypted under h_1 or h_2; equivalently, whether $\mathsf{id} \le \mathsf{id}_2$ or $\mathsf{id} > \mathsf{id}_2$. We cannot allow this information to be leaked if $\mathsf{id} \notin S_{\mathsf{R}}$. To fix this issue we assume the hash-encryption scheme is *anonymous*, meaning that, roughly, a random ciphertext leaks no information about the underlying hash value. This property was defined in [BLSV18] for achieving anonymous IBE. The use of anonymous hash encryption does not resolve the issue completely yet. For concreteness, suppose $\mathsf{id} < \mathsf{id}_1$. This means that $\{\mathsf{ct}'_{i,b}\}$ is encrypted under h_1, and so by decrypting $\{\mathsf{ct}'_{i,b}\}$ using $z' = (\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$, the receiver will obtain meaningful labels, evaluating the garbled circuit $\widetilde{\mathsf{C}}_0$ to $\perp$ (rightly so, because $\mathsf{id} \notin S_{\mathsf{R}}$). On the other hand, if the receiver tries decrypting $\{\mathsf{ct}'_{i,b}\}$ using $z'' = (\mathsf{id}_3, \mathsf{id}_4, \mathsf{id}_3)$ which is not a pre-image of h_1, then the resulting labels will be meaningless, evaluating $\widetilde{\mathsf{C}}_0$ to junk. This leaks which path is the right binary search path, giving information about id. To fix this issue, we change the circuit F so that if $\mathsf{id} \notin S_{\mathsf{R}}$, then decryption along any path will result in a random value. Specifically, sample two random values r and r', let $\mathsf{F}[\mathsf{id}, r, r'](\mathsf{id}', \mathsf{id}'', *)$ return r if $\mathsf{id} \notin \{\mathsf{id}', \mathsf{id}''\}$ and r' otherwise. We will also include r in the clear in psi_2. Now the receiver can check decryption along which path (if any) yields r; in which case, the receiver can determine the intersection identity. To argue security, if we use anonymous garbled circuits [BLSV18], then we can argue if $\mathsf{id} \notin S_{\mathsf{R}}$, then psi_2 is pseudorandom to the receiver. Arguing this formally (especially for the general case) is non-trivial, requiring a delicate formulation of hybrids.

Receiver's security? The receiver's hash $\mathsf{hr}_{\mathsf{root}}$ is computed deterministically from S_{R}, so it cannot be secure. But this is easy to fix: On the leaf level we append the identities with random values and only then will perform the Merkle hash.

3.2.2 Reusable Laconic PSI

We now outline our techniques for obtaining laconic PSI in a black-box way, for both semi-honest and malicious cases.

A semi-honestly secure protocol Our starting point is a recent construction of a *one-way function with encryption* from the φ-hiding assumption due to Goyal, Vusirikala and Waters [GVW20], and we remark that similar *accumulator-style* ideas were used before to construct PSI [ADT11]. Since the protocol of [GVW20] is "almost" a PSI protocol, we will directly describe the underlying semi-honestly secure PSI based. Assume for a moment that both the receiver's input S_{R} and the sender's input S_{S} are subsets of a polynomially-sized universe $\mathcal{U} = \{1, \ldots, \ell\}$. We will later remove this size restriction on $\mathcal{U}$. We have a common reference string crs which is composed of an RSA modulus $N = PQ$, a uniformly random generator $g \in \mathbb{Z}_N^*$ and pairwise distinct primes $p_1, \ldots, p_\ell$.

For the sake of simplicity, we will assume in this outline that the sender's input set S_{S} is a singleton set $\{w\} \subseteq \mathcal{U}$. The actual protocol will be obtained by running the protocol we will now sketch for every element in the sender's input set. The protocol commences as follows: The receiver first *hashes* its input set into

$$h = g^{r \prod_{i \in S_{\mathsf{R}}} p_i} \mod N,$$

where r is chosen a uniformly chosen random from $[N]$ (and thus $r \bmod \varphi(N)$ is statistically close to uniform). The receiver then sends h to the sender.

The sender, whose input is $S_{\mathsf{S}} = \{w\}$, chooses a uniformly random value $\rho \leftarrow_\$ [N]$ and a uniformly random seed s for a suitable randomness extractor Ext, and computes the values $f \leftarrow g^{\rho p_w}$ and $R \leftarrow \mathsf{Ext}(s, h^\rho)$. It sends s, f and R to the receiver.

The receiver, upon receiving f and R, will check for all elements $i \in S_{\mathsf{R}}$ whether it holds that $R_i \stackrel{?}{=} R$, for $R_i \leftarrow \mathsf{Ext}(s, f^{r \cdot \prod_{j \in S_{\mathsf{R}} \setminus \{i\}} p_j})$. If it finds such an i, it outputs $\{i\}$ as the intersection of S_{R} and S_{S}. Correctness of this protocol follows routinely[8]. by noting that if $w \in S_{\mathsf{R}}$ then

$$f^{r \cdot \prod_{j \in S_{\mathsf{R}} \setminus \{w\}} p_j} = g^{\rho \cdot r \cdot \prod_{j \in S_{\mathsf{R}}} p_j} = h^\rho.$$

Also, note that this scheme is laconic, as the size of the messages exchanged by the parties is independent of the size of the set S_{R}.

Arguing security against a semi-honest sender is also routine, as h is in fact statistically close to a uniformly random group element in $\mathbb{Z}_N^*$. Proving security against a semi-honest receiver is a bit more involved and proceeds via the following hybrid modifications. Let $S_{\mathsf{S}} = \{w\}$ be the sender's input such that $w \notin S_{\mathsf{R}}$. In the first hybrid, we will choose the modulus N such that p_w divides $\varphi(N)$; under the φ-hiding assumption, this change will go unnoticed. Now, via a standard lossiness-argument, we have that $f = g^{\rho p_w}$ loses information about g^ρ, i.e., g^ρ has high min-entropy given f. This means that $h^\rho = g^{\rho r \cdot \prod_{i \in S_{\mathsf{R}}} p_i}$ has also high min-entropy as $w \notin S_{\mathsf{R}}$ and thus p_w does not divide $r \cdot \prod_{i \in S_{\mathsf{R}}} p_i$ (w.o.p). Consequently, as h^ρ has high min-entropy conditioned on f, in the next hybrid change we can replace $R = \mathsf{Ext}(s, h^\rho)$ with a uniformly random value, incurring only a negligible statistical distance via the extraction property of Ext. In the next hybrid change, we can switch the modulus N back to normal mode, i.e., such that p_w does not divide $\varphi(N)$. But now $f = g^{\rho p_w}$ is statistically close to uniform in $\mathbb{Z}_N^*$. Thus, in the last hybrid change, we can replace f with a uniformly random value in $\mathbb{Z}_N^*$ and get that the view of the receiver is independent of w, as required.

[8]We will not further discuss the small correctness-error of this protocol as our final protocol will not suffer from this defect

For the case that the sender's input S_{S} contains more than a single element, we mount a hybrid argument repeating the above modifications for each element of S_{S}, not in the receiver's set S_{R}.

Large universes The above protocol has the drawback that the size of the common reference string crs depends linearly on the size of the universe $\mathcal{U}$, which is highly undesirable. There is a standard way of overcoming this issue: Instead of explicitly listing all the primes p_i in crs, we will describe them implicitly via a pseudorandom function (PRF).[9] For this purpose, we need a PRF which maps into the set of primes of a certain size. This can e.g. be achieved by using rejection sampling: we first sample $y \leftarrow F_k(x|i)$ (starting with $i = 1$) and check if y is a prime number. If it is, we output y; else, we increment i until a prime is hit. Under standard number-theoretic assumptions, this process finds a prime after a logarithmic number of steps. One small issue is that, in the above security proof, we need to replace one of the primes with a prime provided by the φ-hiding experiment. We resolve this issue by making the PRF programmable in one point, e.g., by setting $F_{k,k'}(x|i) = F'_k(x|i) \oplus k_i$ for a PRF F', $k' = (k_1, \ldots, k_\xi)$ and a suitable choice of ξ.

A first attempt at malicious sender security Our protocol thus far, however, offers no security against a malicious sender. The main issue is that a corrupted sender may choose the values f and R arbitrarily, and further, there is no mechanism for a simulator against a malicious sender to extract the senders input w. Of course, this protocol can be made secure against malicious senders by letting the sender prove via a general-purpose NIZK proof that it follows the semi-honest protocol correctly. This however would necessitate making a non-black-box use of our semi-honest laconic PSI protocol, contrary to our goal of achieving a fully black-box protocol.

Re-inspecting the above protocol, we have not made full use of the fact that the extracted string R is uniformly random. Our first idea to make the sender extractable is to make better use of R. Instead of sending R in the plain, we will use R as random coins for a public key encryption (PKE) scheme to encrypt the sender's input w. More concretely, we will modify the above protocol as follows. We include a public key pk of a PKE scheme in the common reference string crs and, instead of having the sender include R in the plain in its message to the receiver, it will include a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, i; R)$. We also need to modify the procedure of the receiver. The receiver will recover R_i as before, but will now use R_i to *re-encrypt* the index i, that is, for each $i \in S_{\mathsf{R}}$ it will compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, i; R_i)$.

First notice that, as a side bonus, this modification makes our laconic PSI scheme perfectly correct, given that the PKE scheme is perfectly correct, as now ct_i uniquely specifies the element i.

In terms of security, we first observe that this modification does not harm security against a semi-honest receiver given that the PKE scheme is IND-CPA secure. In the above sketch of a security proof, we have argued that, if w is not in the set S_{R}, then R is uniformly random from the view of the receiver. This means now that ct is a freshly encrypted ciphertext, using fresh random coins (independent of ρ). Moreover, we can use IND-CPA security of the PKE to replace ct with encryption of 0, and then continue as above to argue security against a semi-honest receiver.

[9]We remark that we use a PRF, not because we want uniform outputs, but to implicitly define the set of primes. A similar trick was used in [BGI16].

To establish security against a malicious sender, we would like to argue as follows. The simulator can now generate the public key pk in crs together with a secret key sk. Given a message (s, f, ct) by a malicious sender, the simulator can recover the set element w by decrypting the ciphertext ct using sk. At a first glance, this seems to provide us with security against malicious senders. And indeed, the simulator will recover all elements for which the receiver would have declared to be in the intersection. There is a grave issue, however: The simulator has no means of detecting whether the honest receiver would actually have succeeded in re-encrypting the index i. In other words, the malicious sender can make the simulator *false positives*, such that the simulator declares an element i to be in the intersection, whereas an honest receiver would not have.

Switch groups, extract everything! We briefly recall some facts about the Damgård-Jurik cryptosystem [DJ01]. The group $\mathbb{Z}^*_{N^{\xi+1}}$ contains a cyclic subgroup $\mathbb{NR}_N$ of order $\varphi(N)$[10]. Now let $g_0 \in \mathbb{NR}_N$ be a generator of $\mathbb{NR}_N$. Then we can generate the entire group $\mathbb{Z}^*_{N^{\xi+1}}$ by g_0 and $1 + N$, i.e. we can write every $h \in \mathbb{Z}^*_{N^{\xi+1}}$ as $h = g_0^t \cdot (1 + N)^m$ for some $t \in \mathbb{Z}_{\varphi(N)}$ and $m \in \mathbb{Z}_{N^\xi}$. Furthermore, we can efficiently compute discrete logarithms relative to $1 + N$, i.e. if $h = (1 + N)^m$ for an $m \in \mathbb{Z}_{N^\xi}$, then we can efficiently compute m from h. Finally, the decisional composite residue (DCR) assumption in $\mathbb{Z}^*_{N^{\xi+1}}$ states that a random element in $\mathbb{NR}_N$ is indistinguishable from a random element in $\mathbb{Z}^*_{N^{\xi+1}}$. It follows that $g_1 = g_0^{t_1}$ and $g_2 = g_0^{t_2} \cdot (1 + N)$ (for uniformly random $t_1, t_2 \leftarrow_\$ \mathbb{Z}_{\varphi(N)}$) are computationally indistinguishable. Moreover, if $h = g_2^t$ for a $t < N^{\xi-1}$, we can efficiently compute t from h using $\varphi(N)$ as a trapdoor by first computing

$$h^{\varphi(N)} = g_2^{t\cdot\varphi(N)} = \underbrace{g_0^{t\varphi(N)}}_{=1} \cdot (1 + N)^{t\cdot\varphi(N)} = (1 + N)^{t\cdot\varphi(N)} \mod N^{\xi+1},$$

from which we can efficiently compute $t \cdot \varphi(N)$ (as $t \cdot \varphi(N) < N^\xi$) and thus t.

Given this, we will now make the following additional modification to our PSI protocol. Instead of choosing the element g in the common reference string crs to be a random generator of $\mathbb{Z}^*_N$, we choose g to be a random generator of $\mathbb{NR}_N$, where $\mathbb{NR}_N$ is the subgroup of order $\varphi(N)$ in $\mathbb{Z}^*_{N^{\xi+1}}$ (for a sufficiently large but constant ξ). Our first observation is that this does not affect the security proof in the case of a semi-honest receiver, since $\mathbb{NR}_N$ is still a cyclic group of order $\varphi(N)$ and the above argument using the φ-hiding assumption works analogously in this group.

Assume for a moment we had a mechanism which ensures that the group element f in the sender's message is of the form $f = g^a$ for an $a < N^{\xi-1}$. We can then argue security against a malicious sender as follows: First, we make a hybrid change and choose the element g in the common reference string like g_2 above, i.e. we choose $g = g_0^t(1+N)$; under the DCR assumption, this change goes unnoticed.

[10]Note that $\mathbb{NR}_N$ is not a cyclic group and we only assume this here for simplicity. Actually, if we choose N as a product of two safe primes, then we could find a cyclic subgroup $\mathbb{J}_N$ which is the group of elements with Jacobi symbol 1, and its subgroup $\mathbb{T}_N$ composing of N^ξ-th powers of $\mathbb{J}_N$ has order $\varphi(N)/2$. Namely, just replace the group pair $(\mathbb{Z}^*_{N^{\xi+1}}, \mathbb{NR}_N)$ with $(\mathbb{J}_N, \mathbb{T}_N)$ to fix this issue. Please refer to Section 1.4 and Section 3.6 for details.

Now, given that $f = g^a$ for an $a < N^{\xi-1}$ and using $\varphi(N)$ as a trapdoor, the simulator can efficiently compute a from f as described above. Since it can also recover the index w from the ciphertext ct as described above, it can now check if a is of the form $a = \rho \cdot p_w$. If so, it recovers ρ and performs the same re-encryption test for ct which the real receiver would perform. This makes the simulation indistinguishable from the real experiment.

3.2.3 DV-NIZK Range Proofs for DJ Ciphertexts

The final component which is missing to make the above argument succeed is a mechanism which ensures that the group element f is true of the form $f = g^a$ for a *small* a. For the sake of generality, we will make the following discussion for general DJ-ciphertexts, that is, ciphertexts of the form $c = h^t \cdot (1+N)^a$ (where $h = g_1^z$ is the public key). If we can show that such a ciphertext encrypts a small value a, proving that $f = g^a$ and $c = h^t \cdot (1+N)^a$ for the same a can be efficiently proven via a standard hash-proof system (HPS) [CS02].

First, we observe that, to show that $c = h^t \cdot (1+N)^a$ encrypts a value $a < 2^k$ for some parameter k, it suffices to prove that some ciphertexts $c_0, \ldots, c_{k-1}$ encrypt *bits* $b_1, \ldots, b_{k-1}$. Assume for now we had a DV-NIZK protocol Π to prove that the ciphertexts $c_0, \ldots, c_{k-1}$ all just encrypt bits. The prover can convince the verifier as follows c encrypts a value $a < 2^k$. First the prover encrypts bit b_i in a ciphertext c_i and sets $c' = \prod_{i=0}^{k-1} c_i^{2^i}$ (it is not hard to see that c' encrypts a). Now, the prover uses Π to convince the verifier that $c_0, \ldots, c_{k-1}$ indeed encrypt bits. Furthermore, it can use a standard HPS to prove that c and c' indeed encrypt the same value. Zero-knowledge follows routinely. To see that this protocol is sound, observe that if the c_i indeed encrypt bits, then c' must encrypt a value bounded by 2^k.

A DV-NIZK proof system for ciphertext equality across different encryption schemes

Alas, we do not know of a black-box DV-NIZK which proves that DJ ciphertexts encrypt bits. However, for the pairing-based Boneh-Goh-Nissim (BGN) cryptosystem [BGN05], such a proof system was constructed by Groth, Ostrovsky and Sahai [GOS06]. Consequently, if we could prove in a black box way that a BGN ciphertext encrypts the same value as a DJ ciphertext we would be done.

Recall that, in the BGN cryptosystem, public keys are of the form (G, H), where G and H generators of subgroups of a composite-order pairing group $\mathbb{G}$. BGN ciphertexts are of the form $C = G^m H^r$, where m is the encrypted message and r are random coins.

Our final contribution is a DV-NIZK proof system which allows us to prove that a DJ ciphertext and a BGN ciphertext encrypt the same value.

To simplify the description of our prove system, assume we have BGN public keys $(G, H_1), \ldots, (G, H_\ell)$, i.e. each key sharing the same G but having fresh and random H_i, and an element H_0. Furthermore, assume that we have DJ public keys $h_1, \ldots, h_\ell$, and an element h_0. We will assume that both sequences of keys are in a public setup, together with the elements H_0, h_0.

Suppose further that we have BGN ciphertexts $C_1, \ldots, C_\ell$, where $C_i = G^{m_i} H_i^r$, i.e., all ciphertexts use the same random coins r but encrypt possibly different bits m_i.[11] As mentioned above, using the NIZK scheme from [GOS06], we can prove that the ciphertexts $C_i = G^{m_i} H_i^r$ are indeed well-formed and that $m_i \in \{0,1\}$. Moreover, we have $C_0 = H_0^r$, which can be proven well-formed using a standard hash proof system (HPS) [CS02].

Assume further that we are given DJ ciphertexts $c_1, \dots, c_\ell$, where $c_i = h_i^t \cdot (1+N)^{m_i'}$, i.e., again the ciphertexts share the same random coins t.[12] Moreover, assume that we have a value h_0^r exactly as above. We want to prove that it holds for all $i \in [\ell]$ that $m_i = m_i'$. Our DV-NIZK proof system for equality of BGN and DJ ciphertexts now proceeds roughly as follows:

- The verifier starts by sampling a uniformly random binary string $\sigma \leftarrow\$ \{0,1\}^\ell$ and computes $F = H_0^A \prod H_i^{\sigma_i} \in \mathbb{G}$ and $f = h_0^\alpha \prod h_i^{\sigma_i} \in Z^*_{N^{\xi+1}}$, for uniformly random values A, α. It sends $\mathsf{crs} = (F, f)$ to the prover and keeps σ as the designated verifier key.

- The prover is given ciphertexts $C_1, \dots, C_\ell$ and $c_1, \dots, c_\ell$ with $C_i = G^{m^i} H_i^r$ and $c_i = h_i^t(1+N)^{m_i}$, and the values $C_0 = H_0^r$ and $c_0 = h_0^t$. It computes $K = F^r G^\tau$ and $k = f^t(1+N)^\tau$ where τ is sampled according to a distribution which is wide enough to drown the m_i, but short enough such that it is bounded by N. The proof π is consists of (K, k).

- The verifier, given the proof $\pi = (K, k)$, computes the discrete log y (in base $(1+N)$) of $k^{-1} c_0^\alpha \prod_{i=1}^\ell c_i^{\sigma_i}$ and checks if $G^y = K^{-1} C_0^A \prod_{i=1}^\ell C_i^{\sigma_i}$.

For completeness, note that

$$\begin{aligned} k^{-1} c_0^\alpha \prod c_i^{\sigma_i} &= \left(h_0^\alpha \prod h_i^{\sigma_i}\right)^{-t} (1+N)^{-\tau} \left(h_0^t\right)^\alpha \prod \left(h_i^t (1+N)^{m_i}\right)^{\sigma_i} \\ &= (1+N)^{\sum \sigma_i m_i - \tau}, \end{aligned}$$

from which the verifier can recover $y = \sum \sigma_i m_i - \tau$. Moreover

$$L = K^{-1} C_0^A \prod C_i^{\sigma_i} = \left(H_0^A \prod H_i^{\sigma_i}\right)^{-r} G^{-\tau} \left(H_0^r\right)^A \prod (H_i^r G^{m_i})^{\sigma_i} = G^{\sum \sigma_i m_i - \tau}$$

and thus $G^y = L$.

The zero-knowledge property can be established by noting that the term τ statistically drowns $\sum_i \sigma_i m_i$.

We argue as follows to prove *reusable statistical soundness* (or simulation soundness). First note that σ is statistically hidden, given $F = H_0^A \prod H_i^{\sigma_i}$ and $f = h_0^\alpha \prod h_i^{\sigma_i}$, by the uniform values A, α. We need to show that if there is an index i for which $m_i \neq m_i'$, then the verifier will reject with high probability, irrespective of the (adversarial) choices of τ, τ' (which are not necessarily short)[13]. It follows from the

[11] Via a standard rerandomization argument we can show that reusing the same random coins across different keys does not harm CPA security.

[12] Same as above.

[13] We assume that the verifier rejects if it fails to compute the discrete logarithm of $k^{-1} \prod d_i^{\sigma_i}$.

above description that the verifier accepts proof of the condition

$$\sum \sigma_{i,j} m_i - \tau_j \mod n = \left(\sum \sigma_{i,j} m'_i - \tau'_j \bmod N^{\xi} \right) \mod n$$

is satisfied, where n is the order of the subgroup of $\mathbb{G}$ generated by G. In the main body we will show that, given that $n > N^{\xi}$, this condition will be violated with probability $\approx 1/2$ if there exists an index i for which $m_i \neq m'_i$. By repeating the protocol λ times, we achieve negligible soundness error.

3.2.4 Labeled Laconic PSI and Laconic OT

Our laconic PSI construction can be easily extended into a labelled laconic PSI, in which the receiver also learns labels associated with set elements in the intersection. To achieve this, we simply use an extractor with an output size twice as large: the first half is used as above to perform the re-encryption step; the other half is used as a one-time pad to encrypt the corresponding label. It is easy to see that the receiver can only recover the labels for the elements within the intersection since the security proof follows the same blueprint as before.

We also build a LOT using the same ideas as above. The receiver commits to a database $D \in \{0,1\}^{\Gamma}$ by computing $h = g_0^{r \prod_{i=1}^{\Gamma} e_{i,D_i}} \mod N^{\xi+1}$, where each prime $e_{i,b}$ is the output of a PRF (just as before). The sender computes $f_j = g_0^{\rho_j e_{L,j}}$, $F_j = g_1^{\rho_j e_{L,j}} (1+N)^{\rho_j e_{L,j}}$ for each $j \in \{0,1\}$, together with a range proof. Moreover, he encrypts each message as $\mathsf{ct}_j = k_j \oplus m_j$ where $k_j \leftarrow \mathsf{Ext}(s_j, h^{\rho_j})$. Again, security follows the same reasoning as above. Our LOT protocol is the first one to provide security against a malicious sender while incurring communication complexity independent of the size of D.

3.3 Definitions

Laconic Private Set Intersection. An ℓPSI is a two-round protocol that implements a PSI functionality and has special compactness properties.

Definition 3.3.1. A ℓPSI scheme $\mathsf{LPSI} = (\mathsf{GenCRS}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ is defined as follows:

- $\mathsf{GenCRS}(1^{\lambda})$: Takes as input a security parameter 1^{λ}, and outputs a common reference string crs.
- $\mathsf{R}_1(\mathsf{crs}, S_R)$: Takes as input a crs and a set S_R. It outputs a first PSI message psi_1 and a state st.
- $\mathsf{S}(\mathsf{crs}, S_S, \mathsf{psi}_1)$: Takes as input a crs, a set S_S and a first PSI message psi_1. It outputs a second PSI message psi_2.
- $\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$: Takes as input a crs, a state st and a second message psi_2. It outputs a set $\mathcal{I}$.

We require the following properties.

- **Correctness**: The protocol satisfies PSI correctness in the standard sense.

- **Efficiency Requirements.** There exists a fixed polynomial poly such that the length of psi_1 and the running time of S are at most $\mathsf{poly}(\lambda, \log |S_R|)$.

For malicious security, we work in the standard UC-framework [Can01] that allows us to prove the security of protocols under arbitrary composition with other protocols.

We present the (reusable) PSI ideal functionality.

Reusable PSI functionality. The functionality $\mathcal{F}_{\mathsf{rPSI}}$ is parametrized by a universe $\mathcal{U}$ and works as follows:

- **Setup phase.** R sends $(\mathsf{sid}, S_{\mathsf{R}})$ to $\mathcal{F}_{\mathsf{rPSI}}$ where $S_{\mathsf{R}} \subseteq \mathcal{U}$. It ignores future messages from R with the same sid.

- **Send phase.** S sends $(\mathsf{sid}, i, S_{\mathsf{S}} \subseteq \mathcal{U})$ to $\mathcal{F}_{\mathsf{rPSI}}$. $\mathcal{F}_{\mathsf{rPSI}}$ sends $(\mathsf{sid}, i, S_{\mathsf{R}} \cap S_{\mathsf{S}})$ to R. It ignores future messages from S with the same sid and $i \in \mathbb{N}$.

3.4 Semi-Honest Laconic Private Set Intersection from CDH/LWE

In this section, we show how to realize semi-honest ℓPSI from CDH/LWE. Our construction is non-black-box, making use of garbled circuits. This leads to the first feasibility result based on CDH, and an alternative LWE construction to that of [QWW18].

Our construction makes use of hash encryption schemes in conjunction with garbled circuits, which we review below.

Definition 3.4.1 (Hash Encryption [DG17b, BLSV18]). A hash encryption scheme $\mathsf{HE} = (\mathsf{HGen}, \mathsf{Hash}, \mathsf{HEnc}, \mathsf{HDec})$ is defined as follows.

- $\mathsf{HGen}(1^\lambda, n)$: Takes as input a security parameter 1^λ and an input size n and outputs a hash key p.

- $\mathsf{Hash}(\mathsf{p}, z)$: Takes as input a hash key p and $z \in \{0,1\}^n$, and deterministically outputs $h \in \{0,1\}^\lambda$.

- $\mathsf{HEnc}(\mathsf{p}, h, \{m_{i,b}\}_{i\in[n],b\in\{0,1\}}; \{r_{i,b}\})$: Takes as input a hash key p, a hash output h, messages $\{m_{i,b}\}$ and randomness $\{r_{i,b}\}$, and outputs $\{\mathsf{cth}_{i,b}\}_{i\in[n],b\in\{0,1\}}$. We write it shortly as $\{\mathsf{cth}_{i,b}\}$. Overloading notation, each ciphertext $\mathsf{cth}_{i,b}$ is computed as $\mathsf{cth}_{i,b} = \mathsf{HEnc}(\mathsf{p}, h, m_{i,b}, (i, b); r_{i,b})$.

- $\mathsf{HDec}(z, \{\mathsf{cth}_{i,b}\})$: Takes as input a hash input z and $\{\mathsf{cth}_{i,b}\}$ and outputs n messages $(m_1, \ldots, m_n)$.

We require correctness meaning that for the variables above, $(m_1, \ldots, m_n) = (m_{1,z[1]}, \ldots, m_{n,z[n]})$. We define two notions of security.

- **Semantic Security**: Given $z \in \{0,1\}^n$, no adversary can distinguish between encryptions of messages made to indices $(i, \bar{z}_i)$. For any PPT $\mathcal{A}$, sampling $\mathsf{p} \leftarrow_\$ \mathsf{HGen}(1^\lambda, n)$, if $(z, \{m_{i,b}\}, \{m'_{i,b}\}) \leftarrow_\$ \mathcal{A}(\mathsf{p})$ and if $m_{i,z[i]} = m'_{i,z[i]}$ for all $i \in [n]$, then $\mathcal{A}$ cannot distinguish between $\mathsf{HEnc}(\mathsf{p}, h, \{m_{i,b}\})$ and $\mathsf{HEnc}(\mathsf{p}, y, \{m'_{i,b}\})$, where $h := \mathsf{Hash}(\mathsf{p}, z)$.

- **Anonymous Semantic Security**: For a random $\{m_{i,b}\}$ with equal rows (i.e., $m_{i,0} = m_{i,1}$), the output of $\mathsf{HEnc}(\mathsf{p}, h, \{m_{i,b}\})$ is pseudorandom even in the presence of the hash input. Formally, for any $z \in \{0,1\}^n$, sampling $\mathsf{p} \leftarrow_\$ \mathsf{HGen}(1^\lambda, n)$, $h := \mathsf{Hash}(\mathsf{p}, z)$, and sampling $\{m_{i,b}\}$ uniformly at random with the same rows, then $v := (\mathsf{p}, z, \mathsf{HEnc}(\mathsf{p}, h, \{m_{i,b}\}))$ is indistinguishable from another tuple in which we replace the hash-encryption component of v with a random string.

We have the following results from [BLSV18, GGH19].

Lemma 3.4.1. *Assuming CDH/LWE there exists anonymous hash encryption schemes, where $n = 3\lambda$ (i.e., $\mathsf{Hash}(\mathsf{p}, \cdot) \colon \{0,1\}^{3\lambda} \mapsto \{0,1\}^\lambda$).*[14] *Moreover, the hash function* Hash *satisfies robustness in the following sense: for any input distribution on z which samples at least 2λ bits of z uniformly at random,* $(\mathsf{p}, \mathsf{Hash}(\mathsf{p}, z))$ *and* (p, u) *are statistically close, where* $\mathsf{p} \leftarrow_\$ \mathsf{HGen}(1^\lambda, 3\lambda)$ *and* $u \leftarrow_\$ \{0,1\}^\lambda$.

We also review the notion of garbled circuits and the anonymous property, as defined in [BLSV18].

Definition 3.4.2 (Garbled Circuits)**.** A garbling scheme for a class of circuits $\{\mathsf{C} \colon \{0,1\}^n \mapsto \{0,1\}^m\}$ consists of $(\mathsf{Garb}, \mathsf{Eval}, \mathsf{Sim})$ satisfying the following.

- **Correctness**: for all $\mathsf{C} \in \mathcal{C}$, $\mathsf{msg} \in \{0,1\}^n$, $\Pr[\mathsf{Eval}(\tilde{\mathsf{C}}, \{\mathsf{lb}_{i,\mathsf{msg}[i]}\}) = \mathsf{C}(\mathsf{msg})] = 1$, where $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(1^\lambda, \mathsf{C})$.

- **Simulation Security**: For any $\mathsf{C} \in \mathcal{C}$ and $\mathsf{msg} \in \{0,1\}^n$: $(\tilde{\mathsf{C}}, \{\mathsf{lb}_{i,\mathsf{msg}[i]}\}) \overset{c}{\equiv} \mathsf{Sim}(1^\lambda, \mathsf{C}(\mathsf{msg}))$, where $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(1^\lambda, \mathsf{C})$.

- **Anonymous Security** [BLSV18]: For any $\mathsf{C} \in \mathcal{C}$, choosing $y \leftarrow_\$ \{0,1\}^m$, the output of $\mathsf{Sim}(1^\lambda, y)$ is pseudorandom.

Lemma 3.4.2 ([BLSV18])**.** *Anonymous garbled circuits can be built from one-way functions (OWFs).*

Notation on Hash Encryption. Throughout this section we assume $\mathsf{Hash}(\mathsf{p}, \cdot) \colon \{0,1\}^n \mapsto \{0,1\}^\lambda$, where $n = 3\lambda$. We use $\{\mathsf{lb}_{i,b}\}$ to define a sequence of pairs of labels, where (throughout this section) $i \in [n]$ and $b \in \{0,1\}$. For $r := \{r_{i,b}\}$ we let $\mathsf{HEnc}(\mathsf{p}, h, \{\mathsf{lb}_{i,b}\}; r)$ denote the ciphertexts $\{\mathsf{cth}_{i,b}\}$, where $\mathsf{cth}_{i,b} = \mathsf{HEnc}(\mathsf{p}, h, \mathsf{lb}_{i,b}, (i,b); r_{i,b})$. We further overload the notation as follows. We use $\{\mathsf{lb}_i\}$ to denote a sequence of 3λ elements. For $r := \{r_{i,b}\}$ we let $\mathsf{HEnc}(\mathsf{p}, h, \{\mathsf{lb}_i\}; r)$ denote a hash encryption where both plaintext rows are $\{\mathsf{lb}_i\}$; namely, the ciphertexts $\{\mathsf{cth}_{i,b}\}$, where $\mathsf{cth}_{i,b} = \mathsf{HEnc}(\mathsf{p}, h, \{m_{i,b}\}; r_{i,b})$, where $m_{i,0} = m_{i,1} = \mathsf{lb}_i$, for all i.

[14] We note that the CDH construction of [BLSV18] satisfies a weaker notion of anonymity, in which only some part of the ciphertext is pseudorandom. But for ease of presentation we keep the notion as is, and remark that our ℓPSI construction works also with respect to that weaker notion.

Tree Terminology. Throughout this section we work with full binary trees. The depth of a tree is the length of a root-leaf path. We call the leaf level 0, the level above it level one, and so on. We order the root-leaf paths from left to right; namely, the path from the root to the leftmost leaf node is the first root-leaf path, and the path from the root to the rightmost leaf node is the 2^dth root-leaf path, where d is the depth. Each node has an associated hash value, computed based on values associated with its children. Thus, when representing a root-leaf path, we include both children of each branching intermediate node.

Sender's Set Size is One. We assume without loss of generality that the sender holds a single element. For the general case where the sender may have multiple elements, we reuse the first message of the receiver for each element in the sender's set. The overall running time of the sender will only scale with its own set size, and not with the receiver's set size.

Construction 3.4.1 (ℓPSI Construction). *We require the following ingredients in our ℓPSI Construction.*

1. *A hash encryption scheme* $\mathsf{HE} = (\mathsf{HGen}, \mathsf{Hash}, \mathsf{HEnc}, \mathsf{HDec})$*, where* $\mathsf{Hash}(\mathsf{p}, \cdot) \colon \{0,1\}^{3\lambda} \mapsto \{0,1\}^{\lambda}$.
2. *A garbling scheme* $\mathsf{GS} = (\mathsf{Garb}, \mathsf{Eval}, \mathsf{Sim})$.
3. *Circuits* F *and* V*, as well as procedure* $\mathsf{DecPath}$*, defined in Table 3.1.*

We assume the elements of the receiver and the sender are strings in $\{0,1\}^{\lambda}$*. We refer to each element as an identity. Build* $(\mathsf{GenCRS}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ *as follows.*

$\mathsf{GenCRS}(1^{\lambda})$*:* *Return* $\mathsf{crs} \leftarrow_{\$} \mathsf{HGen}(1^{\lambda}, 3\lambda)$.

$\mathsf{R}_1(\mathsf{crs}, S_{\mathsf{R}})$*:* *Assume* $|S_{\mathsf{R}}| = 2^d$*. (With small tweaks the same construction works if* S_{R} *is not a power of two.)*

- *Parse* $\mathsf{crs} := \mathsf{p}$*. Let* $n := 2^d$*, and sort* $S_{\mathsf{R}} := \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$*, where* $\mathsf{id}_i < \mathsf{id}_{i+1}$ *for all* i*. Populate the leaf node values as follows. For each* $\mathsf{id}_i \in S_{\mathsf{R}}$*, sample* $x_i, x_i' \leftarrow_{\$} \{0,1\}^{\lambda}$*, and let* $h_i^{(0)} := \mathsf{Hash}(\mathsf{p}, \mathsf{id}_i, x_i, x_i')$*. Set* $\mathbb{H}[v_i^{(0)}] := h_i^{(0)}$ *and* $\mathsf{ID}[v_i^{(0)}] := \mathsf{id}_i$*.*

  1. *For* $w \in [d]$*, populate the values for the nodes at level* w *as follows. Informally, the hash value for each node is the hash of the concatenation of its left child, and right child, and the largest identity value under its left child. Formally, noting we have* 2^{d-w} *nodes on level* w*, for* $j \in [2^{d-w}]$*, set* $h_j^{(w)} := \mathsf{Hash}(\mathsf{p}, (h_{2j-1}^{(w-1)}, h_{2j}^{(w-1)}, \mathsf{id}_{[j,w]}))$*, where* $\mathsf{id}_{[j,w]}$ *denotes the larges leaf identity under the left child of the current node (i.e.,* $\mathsf{id}_{[j,w]} = \mathsf{id}_f$*, where* $f := (2j-1)2^{w-1}$*.) Set* $\mathbb{H}[v_j^{(w)}] = h_j^{(w)}$ *and* $\mathsf{ID}[v_j^{(w)}] = \mathsf{id}_{[j,w]}$*.*
  2. *Set* $\mathsf{psi}_1 := (d, \mathsf{hr}_{\mathsf{root}})$*, where* $\mathsf{hr}_{\mathsf{root}} := h_1^{(d)}$ *(i.e., the root hash value). Set* $\mathsf{st} := (S_{\mathsf{R}}, \{x_i\}, \{x_i'\}, \{v_j^{(w)}\})$ *for all values of* $i \in [n]$*,* $w \in \{0, \ldots, d\}$ *and* $j \in [2^{d-w}]$*.*

Circuit $\mathsf{F}[\mathsf{id}, r, r'](\mathsf{id}', x, x')$:

- **Hardwired:** target identity id and randomness values r and r'.
- **Operation:** Return
$$y := \begin{cases} r & \mathsf{id} = \mathsf{id}' \\ r' & \text{else} \end{cases}$$

Circuit $\mathsf{V}[\mathsf{p}, \mathsf{id}, \{\mathsf{lb}_{i,b}\}, r](h_1, h_2, \mathsf{id}')$:

- **Hardwired:** Hash public parameter p, target identity id, labels $\{\mathsf{lb}_{i,b}\}$, randomness r.
- **Operation:** Return
$$\mathsf{ct} := \begin{cases} \mathsf{HEnc}(\mathsf{p}, h_1, \{\mathsf{lb}_{i,b}\}; r) & \mathsf{id} \leq \mathsf{id}' \\ \mathsf{HEnc}(\mathsf{p}, h_2, \{\mathsf{lb}_{i,b}\}; r) & \text{else} \end{cases}$$

Procedure $\mathsf{DecPath}(\mathsf{pth}, \mathsf{psi}_2)$:

- **Input:** A leaf-root Path pth and ciphertext $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}^{(d)}_{i,b}\})$.
- **Operation:** Parse $\mathsf{pth} := (\underbrace{(\mathsf{id}, x, x')}_{z_0}, \underbrace{(h_0, h'_0, \mathsf{id}_0)}_{z_1}, \ldots, \underbrace{(h_{d-1}, h'_{d-1}, \mathsf{id}_{d-1})}_{z_d}, \mathsf{hr}_{\mathsf{root}})$. For $w \in \{d, \ldots, 1\}$:
  1. Let $\{\mathsf{lb}^{(w)}_i\} := \mathsf{HDec}(\mathsf{z}_w, \{\mathsf{cth}^{(w)}_{i,b}\})$.
  2. Set $\{\mathsf{cth}^{(w-1)}_{i,b}\} := \mathsf{Eval}(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}^{(w)}_i\})$.

  Let $\{\mathsf{lb}^{(0)}_i\} := \mathsf{HDec}(z_0, \{\mathsf{cth}^{(0)}_{i,b}\})$. Return $\mathsf{Eval}(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}^{(0)}_i\})$.

Table 3.1: Circuits F, V and procedure $\mathsf{DecPath}$

$\mathsf{S}(\mathsf{crs}, \mathsf{id}, \mathsf{psi}_1)$:

- *Parse* $\mathsf{psi}_1 := (d, \mathsf{hr}_{\mathsf{root}})$ *and* $\mathsf{crs} := \mathsf{p}$. *Sample* $r, r' \leftarrow_\$ \{0,1\}^\lambda$ *and let* $\mathsf{C}_0 := \mathsf{F}[\mathsf{id}, r, r']$ *(Table 3.1). Garble* $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}^{(0)}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_0)$. *For* $1 \leq w \leq d$
  1. *Sample* r_w *at random, and let* $\mathsf{C}_w := \mathsf{V}[\mathsf{p}, \mathsf{id}, \{\mathsf{lb}^{(w-1)}_{i,b}\}, r_w]$.
  2. *Garble* $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}^{(w)}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_w)$.
- *Let* $\{\mathsf{cth}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}^{(d)}_{i,b}\})$. *Return* $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$.

$\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$*:*

- *Parse* $\mathsf{st} := (S_\mathsf{R}, \{x_i\}, \{x'_i\}, \{v_j^{(w)}\})$, $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$ *and* $S_\mathsf{R} := \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$. *For* $i \in [n]$ *let* $\mathsf{pth}_i := ((\mathsf{id}_i, x_i, x'_i), \ldots, \mathsf{hr}_\mathsf{root})$ *be the i'th leaf-root path in the tree, and let*

  $$r_i := \mathsf{DecPath}(\mathsf{pth}_i, \widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}).$$

  If for a unique index $i \in [n]$, $r_i = r$, *then output* id_i. *Otherwise, output* $\perp$.

Theorem 3.4.3. *Assuming the hash encryption* HE *is anonymous and robust (robustness defined in Lemma 3.4.1), and that the garbling scheme* GS *is anonymous, the* ℓ*PSI protocol of Construction 3.4.1 provides statistical security for the receiver and semi-honest security for the sender. As a result, such* ℓ*PSI protocols can be realized from CDH/LWE.*

Roadmap for the Proof of Theorem 3.4.3. The fact that the protocol provides statistical security for the receiver follows from the robustness of HE. In particular, robustness implies that the $h_i^{(0)}$ values statistically hide S_R. We can continue this to argue that all the first-level hash values (i.e., $h_i^{(1)}$) also hide S_R, and hence, continuing like this, the root hash value $\mathsf{hr}_\mathsf{root}$ statistically hides S_R.

We now prove that the protocol provides sender security against semi-honest receivers. Let id be the sender's input message, and $S_\mathsf{R} := \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ be the receiver's set, where $\mathsf{id}_i < \mathsf{id}_{i+1}$. Assuming $\mathsf{id} \notin S_\mathsf{R}$ we will show that the sender's protocol message is pseudorandom in the receiver's point of view. For simplicity suppose $\mathsf{id} < \mathsf{id}_1$; the general case follows via simple changes, which we will illustrate in Remark 3.4.2. Let

$$\mathsf{pth} := (\underbrace{(\mathsf{id}_1, x_1, x'_1)}_{z_0}, \underbrace{(h_0, h'_0, \mathsf{id}_0)}_{z_1}, \ldots, \underbrace{(h_{d-1}, h'_{d-1}, \mathsf{id}_{d-1})}_{z_d}, \mathsf{hr}_\mathsf{root}) \tag{3.1}$$

be the leaf-root path from leaf id_1 to the root. Note that $\mathsf{hr}_\mathsf{root} = \mathsf{Hash}(\mathsf{p}, z_d)$, and $h_i = \mathsf{Hash}(\mathsf{p}, z_i)$ for all $i \in \{0, \ldots, d-1\}$. Noting that $\mathsf{hr}_\mathsf{root}$ is the receiver's first-round message, we define the following hybrids for the sender's response message.

$\mathbf{Hyb}_0$: The sender's response message psi_2 is formed as in the protocol.

$\mathbf{Hyb}_1$: Sample $r, r' \leftarrow_\$ \{0,1\}^\lambda$. Let $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_i^{(0)}\} \leftarrow_\$ \mathsf{Sim}(\mathsf{F}, r')$. For $1 \leq w \leq d$

1. Sample $\{\mathsf{cth}_{i,b}^{(w-1)}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, h_{w-1}, \{\mathsf{lb}_i^{(w-1)}\})$.
2. Let $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_i^{(w)}\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}_{i,b}^{(w-1)}\})$.

Let $\{\mathsf{cth}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_\mathsf{root}, \{\mathsf{lb}_i^{(d)}\})$. Return $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$.

Lemma 3.4.4. *Hybrids* $\mathbf{Hyb}_0$ *and* $\mathbf{Hyb}_1$ *are indistinguishable.*

$\mathbf{Hyb}_2$: Sample psi_2 at random.

Lemma 3.4.5. *Hybrids* $\mathbf{Hyb}_1$ *and* $\mathbf{Hyb}_2$ *are indistinguishable.*

The above two lemmas establish the sender's security; namely — if $\mathsf{id} \notin S_{\mathsf{R}}$, then the sender's message psi_2 is pseudorandom for the receiver. We prove Lemma 3.4.4 in Section 3.4.1 and Lemma 3.4.5 in Section 3.4.2.

3.4.1 Proof of Lemma 3.4.4

In the following, given two hybrids $\mathbf{Hyb}$ and $\mathbf{Hyb}'$, we use the notation $\mathbf{Hyb} \overset{c}{\equiv} \mathbf{Hyb}'$ to express that the hybrids are computationally indistinguishable.

We define $d+1$ hybrids between $\mathbf{Hyb}_0$ and $\mathbf{Hyb}_1$, and prove their indistinguishability.

For $p \in \{0, \dots, d\}$ we define $\mathbf{Hyb}'_p$ as follows. Under $\mathbf{Hyb}'_p$, we form the first $p+1$ garbled circuits $\widetilde{\mathsf{C}}_0, \dots, \widetilde{\mathsf{C}}_p$ and their corresponding labels honestly as in the real game, and we simulate the rest.

$\mathbf{Hyb}'_p$: Let pth be as in Equation 3.1, and recall that we are assuming $\mathsf{id} < \mathsf{id}_1$. Sample $r, r' \leftarrow_\$ \{0,1\}^\lambda$ and let $\mathsf{C}_0 := \mathsf{F}[\mathsf{id}, r, r']$. Garble $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_{i,b}^{(0)}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_0)$. Let $\{\mathsf{lb}_i^{(0)}\} := \{\mathsf{lb}_{i,z_0[i]}\}$. Do the following:

- For $1 \leq w \leq p$
  1. Sample r_w at random, and let $\mathsf{C}_w := \mathsf{V}[\mathsf{p}, \mathsf{id}, \{\mathsf{lb}_{i,b}^{(w-1)}\}, r_w]$.
  2. Garble $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_{i,b}^{(w)}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_w)$.
  3. If $w = p$ (i.e., last step), let $\{\mathsf{lb}_i^{(w)}\} := \{\mathsf{lb}_{i,z_w[i]}^{(w)}\}$.
- For $p+1 \leq w \leq d$
  1. Sample $\{\mathsf{cth}_{i,b}^{(w-1)}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, h_{w-1}, \{\mathsf{lb}_i^{(w-1)}\})$.
  2. Let $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_i^{(w)}\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}_{i,b}^{(w-1)}\})$.

Let $\{\mathsf{cth}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\})$. Return $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \dots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$.

Lemma 3.4.6. $\mathbf{Hyb}_0 \overset{c}{\equiv} \mathbf{Hyb}'_d$ *and* $\mathbf{Hyb}_1 \overset{c}{\equiv} \mathbf{Hyb}'_0$.

Proof. We first show $\mathbf{Hyb}_1 \overset{c}{\equiv} \mathbf{Hyb}'_0$. Notice that either hybrid may be simulated just by knowing the value of r and the pair $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_i^{(0)}\})$. We let $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\})$ and $(\widetilde{\mathsf{C}}', \{\mathsf{lb}'_i\})$ denote the distribution of this pair in $\mathbf{Hyb}_1$ and $\mathbf{Hyb}'_0$, respectively. We have $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{F}, r')$. As for the other pair, letting $\mathsf{C}_0 := \mathsf{F}[\mathsf{id}, r, r']$ for random r, r'

$$(\widetilde{\mathsf{C}}', \{\mathsf{lb}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_0) \tag{3.2}$$
$$\{\mathsf{lb}'_i\} = \{\mathsf{lb}_{i,z_0[i]}\}, \tag{3.3}$$

where $z_0 = (\mathsf{id}_1, x_1, x'_1)$. By simulation security of garbled circuits

$$(\widetilde{\mathsf{C}}', \{\mathsf{lb}'_i\}) \overset{c}{\equiv} \mathsf{Sim}(\mathsf{F}, \mathsf{C}_0(z_0)) \tag{3.4}$$
$$\overset{c}{\equiv} \mathsf{Sim}(\mathsf{F}, r'). \tag{3.5}$$

Thus, $(r, \widetilde{\mathsf{C}}, \{\mathsf{lb}_i\})\mathsf{com}(r, \widetilde{\mathsf{C}}', \{\mathsf{lb}'_i\})$, proving $\mathbf{Hyb}_1 \overset{c}{\equiv} \mathbf{Hyb}'_0$.

To prove $\mathbf{Hyb}_0 \overset{c}{\equiv} \mathbf{Hyb}'_d$, their only difference lies in how $\{\mathsf{cth}_{i,b}\}$ is sampled: under $\mathbf{Hyb}_0$: $\{\mathsf{cth}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_{i,b}^{(d)}\})$, while under $\mathbf{Hyb}'_d$: $\{\mathsf{cth}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\})$, where recall that $\{\mathsf{lb}_i^{(d)}\} := \{\mathsf{lb}_{i,z_d[i]}^{(d)}\}$. Since $\mathsf{hr}_{\mathsf{root}} = \mathsf{Hash}(\mathsf{p}, z_d)$, by security of the hash encryption $\mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\}) \overset{c}{\equiv} \mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_{i,b}^{(d)}\})$ and the proof is now complete. □

Lemma 3.4.7. *For all* $p \in \{0, \ldots, d-1\}$*,* $\mathbf{Hyb}'_p \overset{c}{\equiv} \mathbf{Hyb}'_{p+1}$.

Proof. We will show that the distribution of $(\widetilde{C}_0, \ldots, \widetilde{C}_{p+1}, \{\mathsf{lb}_i^{(p+1)}\})$ is computationally indistinguishable in the two worlds. This will imply the result because the rest of either hybrid may be formed based solely on the above tuple. To argue the above tuple is indistinguishable across the two hybrids, first notice that the distribution of

$$(\widetilde{C}_0, \{\mathsf{lb}_{i,b}^{(0)}\}, \ldots, \widetilde{C}_p, \{\mathsf{lb}_{i,b}^{(p)}\})$$

is formed exactly the same in $\mathbf{Hyb}'_p$ and $\mathbf{Hyb}'_{p+1}$. The only difference between these two hybrids lies in the way in which the pair $(\widetilde{C}_{p+1}, \{\mathsf{lb}_i^{(p+1)}\})$ is sampled. To ease notation, we let $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\})$ and $(\widetilde{\mathsf{C}}', \{\mathsf{lb}'_i\})$ denote the distribution of this pair in $\mathbf{Hyb}'_p$ and $\mathbf{Hyb}'_{p+1}$, respectively. Formally

1. **Under $\mathbf{Hyb}'_p$:** We form

$$\{\mathsf{cth}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, h_p, \{\mathsf{lb}_i^{(p)}\}) \tag{3.6}$$
$$(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}\}). \tag{3.7}$$

2. **Under $\mathbf{Hyb}'_{p+1}$**: We form

$$\begin{aligned}(\widetilde{\mathsf{C}}', \{\mathsf{lb}_{i,b}\}) &\leftarrow_\$ \mathsf{Garb}(\mathsf{C}_{p+1}) \\ \{\mathsf{lb}'_i\} &:= \{\mathsf{lb}_{i,z_{p+1}[i]}\},\end{aligned}$$

where $\mathsf{C}_{p+1} := \mathsf{V}[\mathsf{p}, \mathsf{id}, \{\mathsf{lb}^{(p)}_{i,b}\}, r_{p+1}]$ and $z_{p+1} = (h_p, h'_p, \mathsf{id}_p)$.

By simulation security of garbled circuits

$$(\widetilde{\mathsf{C}}', \{\mathsf{lb}'_i\}) \overset{c}{\equiv} \mathsf{Sim}(\mathsf{V}, \mathsf{C}_{p+1}(z_{p+1})) \tag{3.8}$$

$$\overset{c}{\equiv} \mathsf{Sim}(\mathsf{V}, \mathsf{HEnc}(\mathsf{p}, h_p, \{\mathsf{lb}^{(p)}_{i,b}\}; r_{p+1})). \tag{3.9}$$

Notice that in Equation 3.9 we use the fact $\mathsf{id} < \mathsf{id}_p$, and so by definition of C_{p+1}, its hardwired labels $\{\mathsf{lb}^{(p)}_{i,b}\}$ will be encrypted under h_p.[15] Now, Equation 3.9 is identical to the right-hand side of Equation 3.7, and thus $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\}) \overset{c}{\equiv} (\widetilde{\mathsf{C}}', \{\mathsf{lb}'_i\})$. The proof is now complete. □

3.4.2 Proof of Lemma 3.4.5

We need to show that $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}^{(d)}_{i,b}\}, r)$ is pseudorandom, where everything is sampled as in $\mathbf{Hyb}_1$. Since $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}^{(0)}_i\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{F}, r')$ by simulation security of the garbled circuit and Lemma 3.4.2 the distribution of $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}^{(0)}_i\})$ is pseudorandom. Recall that for $1 \leq w \leq d$ we have $\{\mathsf{cth}^{(w-1)}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, h_w, \{\mathsf{lb}^{(w-1)}_i\})$ and $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}^{(w)}_i\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}^{(w-1)}_{i,b}\})$. By Lemma 3.4.1 $\{\mathsf{cth}^{(w-1)}_{i,b}\}$ is pseudorandom, and thus by Lemma 3.4.2 $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}^{(w)}_i\}$ is also pseudorandom, for all $0 \leq w \leq d-1$. Finally, since we have $\{\mathsf{cth}^{(d)}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{p}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}^{(d)}_i\})$, by Lemma 3.4.1, $\{\mathsf{cth}^{(d)}_{i,b}\}$ is pseudorandom. The proof is now complete.

Remark. *In the proof of security, we assumed* $\mathsf{id} < \mathsf{id}_1$. *For the general case, we just need to change the active path from that in Equation 3.1 ending in* id_1 *to the path that will end in* id_j*, where* j *is the largest index such that* id *lies between* id_j *and* id_{j+1}*. In case* $\mathsf{id} > \mathsf{id}_n$*, then* $j = n$*.*

3.5 Reusable DV-NIZK Range Proofs for DJ Ciphertexts

In this section, we construct a DV-NIZK scheme for ranges of DJ ciphertexts. The main idea of our construction is the following: the prover proves that a BGN ciphertext [BGN05] is within a certain range (this can be done via the protocol of [GOS06]). Then it proves that the DJ and BGN ciphertexts encrypt the same value.

[15]This is the place where we use the fact that id is less than all values in S_{R}. In the general case, we should change the above distributions accordingly.

We first recall the required cryptosystems used in this section.

BGN CRYPTOSYSTEM. Recall that the BGN cryptosystem [BGN05] is defined over a group $\mathbb{G}$ of order $n = pq$ for primes p, q. The public key is composed by $(\mathbb{G}, n, G, H)$ where G is a generator of $\mathbb{G}$ and H is an element of order p (let $p\mathbb{G}$ be the subgroup of order p). The public key is composed of $(\mathbb{G}, n, G, H)$ and a ciphertext for a message $m \in \{0, 1\}$ is of the form $C = G^m H^t$ for $t \leftarrow\!\!\$\ [n]$.

DAMGÅRD-JURIK CRYPTOSYSTEM. The Damgård-Jurik (DJ) cryptosystem[16] [DJ01] is defined over $\mathbb{Z}^*_{N^{\xi+1}}$ where $N \leftarrow\!\!\$\ \mathsf{RSA}(\lambda)$. The public key is formed by (N, ξ, g, h) where $g \leftarrow\!\!\$\ \mathbb{T}_N$ and $h = g^x$ for $x \leftarrow\!\!\$\ [N]$. A ciphertext has the form (c_1, c_2) where $c_1 = g^t \bmod N^{\xi+1}$ and $c_2 = h^t(1+N)^m \bmod N^{\xi+1}$ for $t \leftarrow\!\!\$\ [N]$ and $m \in [N^\xi]$.

3.5.1 DV-NIZK Schemes for Linear Languages and for BGN Ciphertexts

We review some basic notions of the hash proof systems framework from [CS02].

Let $\mathbb{X}, \mathbb{L}, \Pi$ be finite abelian groups where $\mathbb{L}$ is a proper subgroup of $\mathbb{X}$. Let $\mathsf{Hom}(\mathbb{X}, \Pi)$ denote the group of all homomorphisms $\varphi : \mathbb{X} \to \Pi$ and let $\mathcal{H}$ be a subgroup of $\mathsf{Hom}(\mathbb{X}, \Pi)$. We call $\mathfrak{G} = (\mathcal{H}, \mathbb{X}, \mathbb{L}, \Pi)$ a group system.

We additionally assume that distinguishing uniformly chosen elements of $\mathbb{L}$ from uniformly chosen elements $\mathbb{X} \setminus \mathbb{L}$ is a NP problem (i.e., distinguishing elements of $\mathbb{L}$ and $\mathbb{X}$ is an instance of a hard subset membership problem [CS02]). We denote by w the witness that states that a given element x is in $\mathbb{L}$ and $\mathcal{R}(x, w)$ the corresponding NP relation.

A group system $\mathfrak{G} = (\mathcal{H}, \mathbb{X}, \mathbb{L}, \Pi)$ is said to be *diverse* if for all $x \in \mathbb{X} \setminus \mathbb{L}$ there exists $\varphi \in \mathcal{H}$ such that $\varphi(\mathbb{L}) = 0$ and $\varphi(x) \neq 0$.

Lemma 3.5.1 (Adapted from [CS02], Theorem 2). *Let $\mathfrak{G} = (\mathcal{H}, \mathbb{X}, \mathbb{L}, \Pi)$ be a diverse group system. Then, there exists a reusable DV-NIZK*

$$\mathsf{NIZK}_\Delta = (\mathsf{NIZK.GenCRS}_\Delta, \mathsf{NIZK.Prove}_\Delta, \mathsf{NIZK.Verify}_\Delta)$$

for the language

$$\mathcal{L} = \{x \in \mathbb{X} : \exists w \text{ s.t. } \mathcal{R}(x, w) = 1\}$$

where $\mathcal{R}$ is the NP *relation that states that $x \in \mathbb{L}$. The scheme fulfills statistical reusable soundness and perfect zero-knowledge.*

It is easy to see that if $\mathbb{X}$ is the product of cyclic groups then, for any proper subgroup $\mathbb{L}$, the group system $\mathfrak{G} = (\mathcal{H}, \mathbb{X}, \mathbb{L}, \Pi)$ is diverse. Thus, there exists a black-box DV-NIZK for the language $\mathcal{L}$.[17]

[16]Here, we present a slightly different variant of the scheme in [DJ01].

[17]To establish that $\mathfrak{G} = (\mathcal{H}, \mathbb{X}, \mathbb{L}, \Pi)$ is a diverse group system we can combine the arguments used in Examples 1 and 2 from Sections 7.4.1 and 7.4.2 in [CS02].

We now review some specific languages which are of the form described above and for which we can obtain efficient black-box DV-NIZK schemes.

In the following, let $N, n \leftarrow\$ \mathsf{RSA}(\lambda)$, $\xi \in \mathbb{N}$ and $\mathbb{G}$ be a group of order n (that is, a BGN group).

DV-NIZK for subgroup membership. First, consider the following language for subgroup membership, which is parametrized by (N, ξ, h)

$$\mathcal{SM}_\Delta = \left\{ h' \in \mathbb{Z}^*_{N^{\xi+1}} : \exists x \in [N] \text{ s.t. } \; h' = h^x \bmod N^{\xi+1} \right\}$$

where $\Delta = (N, \xi, h)$ and $h \in \mathbb{T}_N$. The language allows proving that h' is in the same subgroup as h. In [CS02], a reusable DV-NIZK for discrete log languages with statistical soundness is presented.

We additionally consider the language parametrized by $(\mathbb{G}, n, H)$

$$\mathcal{SM}_{\Delta'} = \left\{ H' \in \mathbb{G} : \exists X \in [n] \text{ s.t. } \; H' = H^X \bmod N^{\xi+1} \right\}$$

where $\Delta' = (\mathbb{G}, n, H)$ and $H \in p\mathbb{G}$.

Lemma 3.5.2 ([CS02] **).** *There exist reusable DV-NIZKs*

$$\mathsf{NIZK}_{\mathcal{SM}_\Theta} = (\mathsf{NIZK.GenCRS}_{\mathcal{SM}_\Theta}, \mathsf{NIZK.Prove}_{\mathcal{SM}_\Theta}, \mathsf{NIZK.Verify}_{\mathcal{SM}_\Theta})$$

for the language $\mathcal{SM}_\Theta$ for $\Theta \in \{\Delta, \Delta'\}$ where $\Delta = (N, \xi, h)$ and $\Delta' = (\mathbb{G}, n, H)$. The scheme fulfills statistical reusable soundness and perfect zero-knowledge.

DV-NIZK for DJ ciphertexts. First, consider the following language for DJ ciphertexts, which is parametrized by $(\{g_i, h_i\}_i, N, \xi)$

$$\mathcal{DJ}_\Delta = \left\{ \{c_{1,i}, c_{2,i}\}_i \in \{\mathbb{Z}^*_{N^{\xi+1}}\}^{2\ell} : \exists (t, \{m_i\}_i) \in [N^\xi]^{\ell+1} \text{ s.t. } \begin{array}{r} c_{1,i} = g_i^t \bmod N^{\xi+1} \\ c_{2,i} = h_i^t (1+N)^{m_i} \bmod N^{\xi+1} \end{array} \right\}$$

for $i \in [\ell]$ where $\Delta = (\{g_i, h_i\}_i, N, \xi)$ and $g_i, h_i \in \mathbb{T}_N$. The language allows proving that h is in the same subgroup as g. In [CS02], a reusable DV-NIZK for discrete log languages with statistically reusable soundness is presented.

Lemma 3.5.3 ([CS02] **).** *There exists a reusable DV-NIZK*

$$\mathsf{NIZK}_{\mathcal{DJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{DJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{DJ}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{DJ}_\Delta})$$

for language $\mathcal{DJ}_\Delta$ where $\Delta = (\{g_i, h_i\}_i, N, \xi)$. The scheme fulfills statistical reusable soundness and perfect zero-knowledge.

DV-NIZK FOR EQUALITY OF DISCRETE LOG. Consider also the following language for the equality of discrete logs.

$$\mathcal{EDL}_\Delta = \left\{ (h_0, h_1) \in \mathbb{Z}^*_{N^{\xi+1}} : \exists t \in [N^\xi] \text{ s.t. } \begin{array}{c} g_0^t = h_0 \bmod N^{\xi+1} \\ g_1^t = h_1 \bmod N^{\xi+1} \end{array} \right\}$$

parametrized by $\Delta = (g_0, g_1, N, \xi)$ where $g_0, g_1 \in \mathbb{Z}^*_{N^{\xi+1}}$. for the equality of discrete logs. Again, the framework of [CS02] can be adapted to obtain an efficient reusable DV-NIZK for this language with statistical reusable soundness.

Lemma 3.5.4 ([CS02]). *There exists a reusable DV-NIZK*

$$\mathsf{NIZK}_{\mathcal{EDL}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{EDL}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{EDL}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{EDL}_\Delta})$$

where $\Delta = (g_0, g_1, N, \xi)$. The scheme fulfills statistical reusable soundness and perfect zero-knowledge.

DV-NIZK FOR EQUALITY OF PLAINTEXTS. Consider the language for the equality of plaintexts in two different DJ ciphertexts

$$\mathcal{EPDJ}_\Delta = \left\{ \{c_i, d_i\}_i \in \{\mathbb{Z}^*_{N^{\xi+1}}\}^4 : \exists(\{t_i\}_i, m) \in [N^\xi]^3 \text{ s.t. } \begin{array}{r} c_1 = g_1^{t_1} \bmod N^{\xi+1} \\ c_2 = h_1^{t_1}(1+N)^m \bmod N^{\xi+1} \\ d_1 = g_2^{t_2} \bmod N^{\xi+1} \\ d_2 = h_2^{t_2}(1+N)^m \bmod N^{\xi+1} \end{array} \right\}$$

for $i = 1, 2$, parametrized by $\Delta = ((g_1, h_1), (g_2, h_2), N, \xi)$ where $g_1, h_1, g_2, h_2 \in \mathbb{T}_N$ for the equality of plaintexts.

Lemma 3.5.5 ([CS02]). *There exists a reusable DV-NIZK*

$$\mathsf{NIZK}_{\mathcal{EPDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{EPDJ}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{EPDJ}_\Delta})$$

where $\Delta = ((g_1, h_1), (g_2, h_2), N, \xi)$. The scheme fulfills statistical reusable soundness and perfect zero-knowledge.

RANGE PROOFS FOR BGN Finally, we consider the language of well-formed BGN ciphertexts encrypting a bit.

$$\mathcal{BGN}_\Delta = \left\{ \{C_i\}_i \in \mathbb{G}^\ell : \exists(t, \{m_i\}) \in [n]^{\ell+1} \text{ s.t. } \begin{array}{c} m_i \in \{0, 1\} \\ C_i = G^{m_i} H_i^t \end{array} \right\}$$

for $i \in [\ell]$, where $\Delta = (\mathbb{G}, n, G, \{H_i\}_{i\in[\ell]})$ and $G, \{H_i\}_{i\in[\ell]} \in \mathbb{G}$.

This is not a linear language and, thus, it cannot be instantiated using the framework of [CS02]. Fortunately, the work of [GOS06] presents an efficient scheme for this language.

Lemma 3.5.6 ([GOS06]). *There exists a reusable DV-NIZK scheme*[18]

$$\mathsf{NIZK}_{\mathcal{BGN}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{BGN}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{BGN}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{BGN}_\Delta})$$

for the language $\mathcal{BGN}$. The protocol has perfect reusable soundness and computational zero-knowledge under the SD assumption.

3.5.2 Equality of Plaintexts in DJ and BGN ciphertexts.

We now show how to prove that a BGN and a DJ ciphertexts encrypt the same value. Consider the following language

$$\mathcal{EQ}_\Delta = \left\{ D_0, h_0, \{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]} : \exists (r, t, \{m_i\}) \text{ s.t. } \begin{array}{c} m_i \in \{0,1\} \\ D_0 = H_0^r \in \mathbb{G} \\ D_i = G^{m_i} H_i^r \in \mathbb{G} \\ c_0 = h_0^t \in \mathbb{Z}^*_{N^{\xi+1}} \\ c_1 = g^t \in \mathbb{Z}^*_{N^{\xi+1}} \\ c_{2,i} = h_i^t (1+N)^{m_i} \in \mathbb{Z}^*_{N^{\xi+1}} \end{array} \right\}$$

where $\Delta = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_{i\in[\ell]})$, such that $G, H_0, \{H_i\}_{i\in[\ell]} \in \mathbb{G}$ and $g, h_0, \{h_i\}_{i\in[\ell]} \in \mathbb{T}_N$.

Construction 3.5.1. *Let $\ell \in \mathbb{Z}$. Let $\Delta = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_{i\in[\ell]})$ be as above, such that $n > N^{\xi+1}$. Let $\beta \in \mathbb{N}$ such that $\lambda/\beta = \mathsf{negl}(\lambda)$, and $N^\xi/2 > \ell\beta$. We require the following ingredients:*

1. *The scheme of Lemma 3.5.6,* $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}} = (\mathsf{NIZK.GenCRS}_{\mathcal{BGN}_{\Delta_1}}, \mathsf{NIZK.Prove}_{\mathcal{BGN}_{\Delta_1}}, \mathsf{NIZK.Verify}_{\mathcal{BGN}_{\Delta_1}})$ *for some $\Delta_1 = (\mathbb{G}, n, G, \{H_i\}_{i\in[\ell]})$.*
2. *The scheme of Lemma 3.5.3,* $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}} = (\mathsf{NIZK.GenCRS}_{\mathcal{DJ}_{\Delta_2}}, \mathsf{NIZK.Prove}_{\mathcal{DJ}_{\Delta_2}}, \mathsf{NIZK.Verify}_{\mathcal{DJ}_{\Delta_2}})$ *for some $\Delta_2 = (\{g, h_i\}_i, N, \xi)$.*
3. *The scheme of Lemma 3.5.3,*

   $$\mathsf{NIZK}_{\mathcal{SM}_\Theta} = (\mathsf{NIZK.GenCRS}_{\mathcal{SM}_\Theta}, \mathsf{NIZK.Prove}_{\mathcal{SM}_\Theta}, \mathsf{NIZK.Verify}_{\mathcal{SM}_\Theta})$$

   for language $\mathcal{SM}_\Theta$ for $\Theta \in \{\Delta_3, \Delta_3'\}$ where $\Delta_3 = (N, \xi, h)$ and $\Delta_3' = (\mathbb{G}, n, H)$.

We present the scheme in full detail.

[18]The scheme presented in [GOS06] is a NIZK scheme and not a DV-NIZK. However, we can view a NIZK as a DV-NIZK where $\mathsf{td} = \perp$.

$\mathsf{GenCRS}_{\mathcal{EQ}_\Delta}(1^\lambda)$:

- *Compute* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{BGN}_{\Delta_1}}(1^\lambda)$ *where* $\Delta_1 = (\mathbb{G}, \{G, H_i\}_{i\in[\ell]})$.
- *Compute* $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{DJ}_{\Delta_2}}(1^\lambda)$ *where* $\Delta_2 = (\{g, h_i\}_{i\in[\ell]}, N, \xi)$.
- *Compute* $(\mathsf{crs}_3, \mathsf{td}_3), (\mathsf{crs}_3', \mathsf{td}_3') \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{SM}_\Theta}(1^\lambda)$ *where* $\Theta \in \{\Delta, \Delta'\}$ *where* $\Delta = (N, \xi, h)$ *and* $\Delta' = (\mathbb{G}, n, H)$.
- *For all* $j \in [\lambda]$, *do the following:*
  - *Sample* $\alpha_j \leftarrow\$ [N/4]$ *and* $A_j \leftarrow\$ [n]$. *For all* $i \in [\ell]$, *sample* $\sigma_{i,j} \leftarrow\$ \{0,1\}$. *Compute* $f_j = h_0^{\alpha_j} \prod_{i=1}^{\ell} h_i^{\sigma_{i,j}} \bmod N^{\xi+1}$ *and* $F_j = H_0^{A_j} \prod_{i=1}^{\ell} H_i^{\sigma_{i,j}}$
- *Output* $\mathsf{crs} = \big(\{F_j, f_j\}_{j\in[\lambda]}, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{crs}_3\big)$ *and* $\mathsf{td} = \big(\{\alpha_j, A_j, \{\sigma_{i,j}\}_{i\in[\ell]}\}_{j\in[\lambda]}, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3\big)$

$\mathsf{Prove}_{\mathcal{EQ}_\Delta}(\mathsf{crs}, x = (D_0, h_0\{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]}), w = (r, t, \{m_{i\in[\ell]}\}))$:

- *Parse* crs *as* $\big(\{F_j, f_j\}_{j\in[\lambda]}, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{crs}_3\big)$.
- *Compute* $\pi_1 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{BGN}_{\Delta_1}}(\mathsf{crs}_1, x_1, w_1)$ *where* $x_1 = \{D_i\}_{i\in[\ell]}$ *and* $w_1 = (r, \{m_i\}_{i\in[\ell]})$.
- *Compute* $\pi_2 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{DJ}_{\Delta_2}}(\mathsf{crs}_2, x_2, w_2)$ *where* $x_2 = \{c_{1,i}, c_{2,i}\}_{i\in[\ell]}$ *and* $w_2 = (t, \{m_i\}_{i\in[\ell]})$.
- *Compute* $\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{SM}_{\Delta_3}}(\mathsf{crs}_3, x_3, w_3)$, *where* $x_3 = c_0$ *and* $w_3 = t$, *and* $\pi_3' \leftarrow \mathsf{NIZK.Prove}_{\mathcal{SM}_{\Delta_3'}}(\mathsf{crs}_3, x_3', w_3')$ *where* $x_3 = D_0$ *and* $w_3' = r$.
- *For all* $j \in [\lambda]$, *do the following: Sample* $\tau_j \leftarrow\$ \Phi_{\mathbb{Z},\beta}$ *and compute* $a_j = f_j^{\,t}(1+N)^{\tau_j} \bmod N^{\xi+1}$. *Compute* $K_j = G^{\tau_j} F_j^r$.
- *Output* $\pi = \big(\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3\big)$.

$\mathsf{Verify}_{\mathcal{EQ}_\Delta}(\mathsf{td}, x, \pi)$:

- *Parse* π *as* $\big(\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3\big)$ *and* td *as* $\big(\{\alpha_j, A_j, \{\sigma_{i,j}\}_{i\in[\ell]}\}_{j\in[\lambda]}, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3\big)$
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{BGN}_{\Delta_1}}(\mathsf{td}_1, x_1, \pi_1)$ *where* $x_1 = \{D_i\}_{i\in[\ell]}$, *output* 0.
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{DJ}_{\Delta_2}}(\mathsf{td}_2, x_2, \pi_2)$ *where* $x_2 = \{c_{1,i}, c_{2.i}\}_{i\in[\ell]}$, *output* 0.
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{SM}_{\Delta_3}}(\mathsf{td}_3, x_3, \pi_3)$ *where* $x_3 = c_0$ *or if* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{SM}_{\Delta_3'}}\big(\mathsf{td}_3, x_3', \pi_3'\big)$ *where* $x_3 = D_0$, *output* 0.
- *For all* $j \in [\lambda]$, *do the following:*

    - *For all* $i \in [\ell]$, *compute* $z_j = a_j^{-1} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} \bmod N^{\xi+1}$. *If there is a* z_j *which is not of the form* $(1+N)^{y_j}$, *output* 0. *Else, recover* y_j.
    - *Compute* $L_j = K_j^{-1} D_0^{A_j} \prod_i D_i^{\sigma_{i,j}}$ *in* $\mathbb{G}$.

- *If there is a* $j \in [\lambda]$ *such that* $G^{y_j} \neq L_j$ *in* $\mathbb{G}$, *output* 0. *Else, output* 1.

Lemma 3.5.7. *The scheme presented in Construction 3.5.1 is complete.*

Proof. Assume $x \in \mathcal{EQ}_\Delta$. Fix a $j \in [\lambda]$. Then,

$$\begin{aligned}
z_j &= a_j^{-1} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_j} \bmod N^{\xi+1} \\
&= \left(h_0^{\alpha_j} \prod_{i=1}^{\ell} h_i^{\sigma_{i,j}} \right)^{-t} (1+N)^{-\tau_j} h_0^{t\alpha_j} \prod_{i=1}^{\ell} h_i^{t\sigma_{i,j}} (1+N)^{m_i \sigma_{i,j}} \bmod N^{\xi+1} \\
&= (1+N)^{\sum_{i=1}^{\ell} m_i \sigma_{i,j} - \tau_j} \bmod N^{\xi+1}
\end{aligned}$$

from which the verifier can recover $y_j = \sum_{i=1}^{\ell} m_i \sigma_{i,j} - \tau_j$. Moreover, $|y_j| < \ell\beta < N^\xi/2 < n/2$. That is, y_j does not wrap around modulo N^ξ nor modulo n.

In addition, we have

$$\begin{aligned}
L_j &= K_j^{-1} D_0^{A_j} \prod_{i=1}^{\ell} D_i^{\sigma_{i,j}} \\
&= \left(H_0^{A_j} \prod H_i^{\sigma_{i,i}} \right)^{-r} G^{-\tau} H_0^{rA_0} \prod_{i=1}^{\ell} (H_i^r G^{m_i})^{\sigma_{i,j}} \\
&= G^{\sum_{i=1}^{\ell} m_i \sigma_{i,j} - \tau_j} = G^{y_j}
\end{aligned}$$

in $\mathbb{G}$. Thus, the proof is accepted as valid because $y_j \mod N^\xi = y_j \mod n$.

□

Lemma 3.5.8. *The scheme presented in Construction 3.5.1 has zero knowledge under the SD assumption.*

Proof. To prove zero knowledge, we construct a simulator $\mathsf{Sim}_{\mathsf{ZK}}$ that creates transcripts which are indistinguishable from the ones outputted by the protocol.

Let Sim_1, Sim_2, Sim_3 and Sim_3' be the zero-knowledge simulators of the schemes $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}}$, $\mathsf{NIZK}_{\mathcal{SM}_{\Delta_3}}$ and $\mathsf{NIZK}_{\mathcal{SM}_{\Delta_3'}}$,respectively (which exist by Lemma 3.5.6, Lemma 3.5.3 and Lemma 3.5.2) The simulator $\mathsf{Sim}_{\mathsf{ZK}}$ works as follows:

- **CRS generation.** It creates a crs exactly as in the real protocol and keeps $\mathsf{td} = \left(\{\sigma_{i,j}\}_{i \in [\ell], j \in [\lambda]}, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3, \mathsf{td}_3' \right)$ to itself.

- Upon receiving an instance $(D_0, h_0\{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]})$, $\mathsf{Sim}_{\mathsf{ZK}}$ first simulates $\pi_1 \leftarrow \mathsf{Sim}_1(\mathsf{td}_1, x_1)$, $\pi_2 \leftarrow \mathsf{Sim}_2(\mathsf{td}_2, x_2)$, $\pi_3 \leftarrow \mathsf{Sim}_3(\mathsf{td}_3, x_3)$ and $\pi_3' \leftarrow \mathsf{Sim}_3(\mathsf{td}_3, x_3')$. Then, it repeats the following for every $j \in [\lambda]$: It samples $\tau_j \leftarrow\$ \Phi_{\mathbb{Z},\beta}$ and computes $a_j = (1+N)^{\tau_j} c_0^{\alpha_j} \prod c_{2,i}^{\sigma_{i,j}} \bmod N^{\xi+1}$ and $K_j = G^{\tau_j} H_0^{A_j} \prod D_i^{\sigma_{i,j}}$.
- It outputs $\pi = (\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3)$.

We now argue that the distributions of the proofs outputted by $\mathsf{Prove}_{\mathcal{EQ}_\Delta}$ and $\mathsf{Sim}_{\mathsf{ZK}}$ are indistinguishable. By the zero-knowledge property of $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}}$, $\mathsf{NIZK}_{\mathcal{SM}_{\Delta_3}}$ and $\mathsf{NIZK}_{\mathcal{SM}_{\Delta_3'}}$ the proofs π_1, π_2, π_3 and π_3' are indistinguishable from the real ones (π_1 is computationally indistinguishable given that the SD assumption holds). So, we just need to analyze the distributions of a_j and K_j.

We start by analyzing the distribution of a_j. First note that,

$$\begin{aligned} a_j^{-1} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} &= c_0^{-\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{-\sigma_{i,j}} (1+N)^{-\tau_j} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} \bmod N^{\xi+1} \\ &= (1+N)^{-\tau_j} \bmod N^{\xi+1} \end{aligned}$$

To see that a_j is indistinguishable from one created in the real-world, note that by Lemma 2.3.1, we have that $\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j \approx_{\mathsf{negl}(\lambda)} \tau_j$, for $\tau_j \leftarrow\$ \Phi_{\mathbb{Z},\beta}$ and $m_i, \sigma_{i,j} \in \{0,1\}$, since $\lambda/\beta = \mathsf{negl}(\lambda)$. Hence,

$$(1+N)^{\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j} \approx_{\mathsf{negl}(\lambda)} (1+N)^{\tau_j}$$

and therefore

$$c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} (1+N)^{\tau_j} \approx_{\mathsf{negl}(\lambda)} f_j (1+N)^{\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j}.$$

An identical argument can be used to show that K_j is indistinguishable from the one created in the real protocol. □

Lemma 3.5.9. *The scheme presented in Construction 3.5.1 has statistical reusable soundness.*

Proof. We first show how the simulator $\mathsf{Sim}_{\mathsf{Snd}}$ simulates the $\mathsf{Verify}_{\mathcal{EQ}_\Delta}(\mathsf{td}, \cdot, \cdot)$ oracle to the adversary. Since we are proving *statistical* reusable soundness, our simulator is allowed to run in exponential time.

$\mathsf{Sim}_{\mathsf{Snd}}$ works as follows:

- **CRS generation.** It generates crs by sampling $f_j \leftarrow\$ \mathbb{T}_N$ and $F_j \leftarrow\$ p\mathbb{G}$.
- Upon receiving a query to the oracle Verify consisting of a statement $x = (D_0, h_0, \{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]})$ together with a proof π from the adversary, it does the following:

1. It brute-forces the statement $(D_0, \{D_i\}_{i\in[\ell]})$ to recover the witness $(r, \{m_i\}_{i\in[\ell]})$, and $(c_0, \{c_{1,i}, c_{2,i}\}_{i\in[\ell]})$ to recover $(t, \{m'_i\}_{i\in[\ell]})$.
2. It parses π as $(\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3)$. It brute-forces a_j to recover $\{\bar{t}_j, \tau'_j\}$, and K_j to recover $(\bar{r}_j, \tau_j)$.
3. If there is an index i such that $(c_{1,i}, c_{2,i})$ are not of the form $c_{1,i} = g^{t_i} \bmod N^{\xi+1}$ and $c_{2,i} = h^{t_i}(1+N)^{m'_i} \bmod N^{\xi+1}$ or D_i is not of the form $D_i = G^{m_i}H_i^r$ in $\mathbb{G}$ where $m_i \in \{0,1\}$, it outputs 0.
4. If there is an index i such that a_i is not of the form $f_i^t(1+N))^{\tau'_i} \bmod N^{\xi+1}$ (meaning that $\bar{t}_i \neq t$), it outputs 0. Moreover, if K is not of the form $G^{\tau_j}F_j^r$ in $\mathbb{G}$ (meaning that $\bar{r}_j \neq r$), output 0.
5. If c_0 is not of the form $h^t \bmod N^{\xi+1}$ or if D_0 is not of the form H_0^r, it outputs 0.
6. If there is $i \in [\ell]$ or $j \in [\lambda]$ such that $m_i \neq m'_i$ or $\tau'_j \neq \tau_j$, it outputs 0. Else, it outputs 1

- Upon receiving the challenge proof (x^*, π^*), it performs the same checks as in steps 3, 4 and 5. If the tests pass, it samples $\sigma_{i,j} \leftarrow\$ \{0,1\}$ and checks if

$$G^{\sum_{i=1}^{\ell}\sigma_{i,j}m'_i - \tau'_j \bmod N^{\xi}} = G^{\sum_{i=1}^{\ell}\sigma_{i,j}m_i - \tau_j}$$

for every $j \in [\lambda]$. It outputs 0 if the test fails, 1 otherwise.

Hyb$_0$: This is the real reusable soundness game.

Hyb$_1$: This game is identical to the previous one except that $\mathsf{Sim}_{\mathsf{Snd}}$ brute-forces the pairs statement/proof (x, π), to recover the witness $(r, \{m_i\}_{i\in[\ell]})$, $(t, \{m'_i\}_{i\in[\ell]})$, and the values $(\tau_j, \bar{r}_j)$ and $\{\bar{t}_j, \tau'_j\}$ from the proof. Finally, it performs the checks in steps 3 and 4.

Claim 3.5.1. *Hybrids* **Hyb**$_0$ *and* **Hyb**$_1$ *are indistinguishable.*

The statistical reusable soundness of the schemes $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}}$, $\mathsf{NIZK}_{\mathcal{SM}_{\Delta_3}}$ and $\mathsf{NIZK}_{\mathcal{SM}_{\Delta'_3}}$ guarantees that D_0, c_0, D_i and $(c_{1,i}, c_{2,i})$ are of the prescribed form, except with negligible probability.

Now fix $j \in [\lambda]$ and assume that $a_j = f_j^{\bar{t}_j}(1+N)^{\tau_j}$. Then, since $a_j^{-1}\prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}}$ must be of the form $(1+N)^{y_j}$, we must have

$$-\bar{t}_j\left(\alpha_j + \sum_{i=1}^{\ell} w_i\sigma_{i,j}\right) + t\left(\alpha_j + \sum_{i=1}^{\ell} w_i\sigma_{i,j}\right) = 0 \bmod \varphi(N)/4$$

where $h_0 = h_i^{w_i}$. Thus $\bar{t}_j = t$.

An identical reasoning can be applied to argue that K_j must be of the form $G^{\tau_j}F_j^r$.

Hyb$_2$. This hybrid is identical to the previous one, except that $\mathsf{Sim}_{\mathsf{Snd}}$ performs the checks in step 6.

Claim 3.5.2. *Hybrids* **Hyb$_1$** *and* **Hyb$_2$** *are indistinguishable.*

The adversary $\mathcal{A}$ is able to distinguish both hybrids if there is a proof which is accepted in hybrid **Hyb$_1$** but rejected **Hyb$_2$** (or vice-versa). That is, suppose that $\mathcal{A}$ outputs $(\{m_i, m'_i\}, \{\tau_j, \tau'_j\})$. Fix j. Then the proof is accepted in **Hyb$_1$** if

$$\sum_{i=1}^{\ell} \sigma_{i,j} m_i - \tau_j \bmod n = \left(\sum_{i=1}^{\ell} \sigma_{i,j} m'_i - \tau'_j \bmod N^{\xi+1} \right) \bmod n. \tag{3.10}$$

Let $e_j = \sum_{i=1}^{\ell} \sigma_{i,j} m_i$ and $d_j = \sum_{i=1}^{\ell} \sigma_{i,j}(m'_i - m_i)$. Then, equation 3.10 can be rewritten as

$$e_j \quad \bmod n = (e_j + d_j - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n.$$

Consider the function $\Gamma_j(z)$ defined as $\Gamma_j(z) = (z - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n$. it is easy to see that $\Gamma_j(z)$ is injective in $\mathbb{Z}_{N^{\xi+1}}$. Let z_1, z_2 be such that

$$\begin{gathered}
\Gamma_j(z_1) = \Gamma_j(z_2) \\
(z_1 - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n = (z_2 - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n \\
(z_1 - \tau'_j \bmod N^{\xi+1}) = (z_2 - \tau'_j \bmod N^{\xi+1}) \\
z_1 \bmod N^{\xi+1} = z_2 - \tau' \bmod N^{\xi+1}
\end{gathered}$$

where the second equivalence holds because $n > N^{\xi+1}$.

Since Γ_j is injective, then we must have

$$\begin{aligned}
e_j \bmod N^{\xi+1} &= e_j + d_j \bmod N^{\xi+1} \\
\sum_{i=1}^{\ell} \sigma_{i,j} m_i \bmod N^{\xi+1} &= \sum_{i=1}^{\ell} \sigma_{i,j} m_i + \sigma_{i,j}(m'_i - m_i) \bmod N^{\xi+1} \\
\sum_{i=1}^{\ell} \sigma_{i,j} m_i \bmod N^{\xi+1} &= \sum_{i=1}^{\ell} \sigma_{i,j} m'_i \bmod N^{\xi+1}
\end{aligned}$$

Assume that there is an index i such that $m_i \neq m'_i$. Then the test will fail with at most $1/2$ probability, for a fixed j. Repeating the process for $j \in [\lambda]$, we get that $m_i = m'_i$, except with negligible probability. Thus $\tau'_j = \tau_j$.

Hybrid $\mathbf{Hyb}_3$. This hybrid is identical to the previous one, except that f_i is chosen uniformly from $\mathbb{T}_N$ and H is chosen uniformly from $p\mathbb{G}$.

Claim 3.5.3. *Hybrids $\mathbf{Hyb}_2$ and $\mathbf{Hyb}_3$ are indistinguishable.*

Since $\alpha_j \leftarrow\!\!\$\ \mathbb{Z}_N$, we can build a hybrid where α_j is sampled from $\mathbb{Z}_N^*$, incurring a difference only in the statistical distance. Moreover, since H_0 is a generator of $p\mathbb{G}$, $H_0^{A_j}$ is uniform in $p\mathbb{G}$. The claim follows.

Claim 3.5.4. *Let $\mathcal{A}$ be any adversary. For hybrid $\mathbf{Hyb}_3$, $\mathcal{A}$ has a negligible advantage.*

Assume that $\mathcal{A}$ outputs (x^*, π^*) where $x^* \notin \mathcal{EQ}_\Delta$. Since $\sigma_{i,j} \leftarrow\!\!\$\ \{0,1\}$, then the proof gets accepted if equation 3.10 is fulfilled. As we have seen before, this happens only with negligible probability.

□

3.5.3 DV-NIZK for Range Proofs of DJ Ciphertexts with Equal Discrete Log

Let $N \leftarrow \mathsf{RSA}(\lambda)$ and $\xi \geq 0$ be a fixed integer. Consider the following language of ranges:

$$\mathcal{REDJ}_\Delta = \left\{ c_1 \in \{\mathbb{Z}^*_{N^{\xi+1}}\}^2 : \exists t \in \{\lceil -N^\xi/2, \ldots, N^\xi/2 \rceil\} \text{ s.t. } \begin{array}{c} t \in [-B, B] \\ c_1 = g^t \bmod N^{\xi+1} \end{array} \right\}$$

which is parametrized by $\Delta = (g, B, N, \xi)$ where $g \in \mathbb{T}_N$, $B \in \mathbb{Z}$, N and ξ.

In the following, we present a DV-NIZK scheme for the language above. The main idea is quite simple: The prover outputs BGN ciphertexts D_i encrypting bits m_i and DJ ciphertexts $(c_{1,i}, c_{2,i})$ that encrypt the same values as D_i (we can prove this using the scheme from the previous section). Then, the prover proves that (c_1, c_2) encrypts the same value as $\left(\prod_{i=0}^{\ell} c_{1,i}^{2^i}, \prod_{i=0}^{\ell} c_{2,i}^{2^i}\right)$. Since DJ is linearly-homomorphic, we conclude that (c_1, c_2) encrypts $m = \sum_{i=0}^{\ell} 2^i m_i \leq 2^{\ell-1}$.

Construction 3.5.2. *Let $\ell \in \mathbb{N}$ and $B = 2^{\ell-1}$ Let*

- $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}} = (\mathsf{NIZK.GenCRS}_{\mathcal{EQ}_{\Delta_1}}, \mathsf{NIZK.Prove}_{\mathcal{EQ}_{\Delta_1}}, \mathsf{NIZK.Verify}_{\mathcal{EQ}_{\Delta_1}})$ *be the scheme in Construction 3.5.1, for some* $\Delta_1 = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_i)$*;*
- $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}} = (\mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_{\Delta_2}}, \mathsf{NIZK.Prove}_{\mathcal{EPDJ}_{\Delta_2}}, \mathsf{NIZK.Verify}_{\mathcal{EPDJ}_{\Delta_2}})$ *be the scheme of Lemma 3.5.5, for some* $\Delta_2 = ((g_1, h_1), (g_2, h_2), N, \xi)$*;*
- $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}} = (\mathsf{NIZK.GenCRS}_{\mathcal{EDL}_{\Delta_3}}, \mathsf{NIZK.Prove}_{\mathcal{EDL}_{\Delta_3}}, \mathsf{NIZK.Verify}_{\mathcal{EDL}_{\Delta_3}})$ *be the scheme of Lemma 3.5.4, for some* $\Delta_3 = (g, h, N, \xi)$.

We now present the scheme in full detail.

$\mathsf{GenCRS}_{\mathcal{REDJ}_\Delta}(1^\lambda)$:

- *Compute* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EQ}_{\Delta_1}}(1^\lambda)$ *where* $\Delta_1 = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_{i\in[\ell]})$.
- *Compute* $(\mathsf{crs}_{2,i}, \mathsf{td}_{2,i}) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_{\Delta_2}}(1^\lambda)$ *where* $\Delta_{2,i} = ((g,h),(g,h_i),N,\xi)$ *for all* $i \in [\ell]$.
- *Compute* $(\mathsf{crs}_{2,0}, \mathsf{td}_{2,0}) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_{\Delta_2}}(1^\lambda)$ *where* $\Delta_{2,0} = ((g,h),(g,h),N,\xi)$.
- *Compute* $(\mathsf{crs}_3, \mathsf{td}_3) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EDL}_{\Delta_3}}(1^\lambda)$ *where* $\Delta_3 = (g, h(1+N), N, \xi)$.
- *Output* $\mathsf{crs} = (\mathsf{crs}_1, \mathsf{crs}_{2,0}, \mathsf{crs}_3, \{\mathsf{crs}_{2,i}\}_{i\in[\ell]})$ *and* $\mathsf{td} = (\mathsf{td}_1, \mathsf{td}_{2,0}, \mathsf{td}_3, \{\mathsf{td}_{2,i}\}_{i\in[\ell]})$.

$\mathsf{Prove}_{\mathcal{REDJ}_\Delta}(\mathsf{crs}, x = c_1, w = t)$:

- *Parse* crs *as* $(\mathsf{crs}_1, \mathsf{crs}_{2,0}, \mathsf{crs}_3, \{\mathsf{crs}_{2,i}\}_{i\in[\ell]})$. *Sample* $r \leftarrow\!\!\$\ \mathbb{Z}_N$ *and* $s \leftarrow\!\!\$\ \mathbb{Z}_N$. *Compute the ciphertexts* $D_i = G^{t_i}H_i^r$, $D_0 = H_0^r$, $d_0 = h_0^s \bmod N^{\xi+1}$ *and* $(d_{1,i}, d_{2,i})$ *where* $d_{1,i} = g^s \bmod N^{\xi+1}$ *and* $d_{2,i} = h_i^s(1+N)^{t_i} \bmod N^{\xi+1}$. *Additionally, compute* $c_2 = h^t(1+N)^t$. *Compute the proof* $\pi_1 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{EQ}_{\Delta_1}}(\mathsf{crs}_1, x_1, w_1)$ *where* $x_1 = (D_0, d_0, \{D_i, d_{1,i}, d_{2,i}\}_{i\in[\ell]})$ *and* $w_1 = (r, s, \{t_i\}_{i\in[\ell]})$.
- *For* $i \in [\ell]$, *sample* $s_i \leftarrow\!\!\$\ \mathbb{Z}_N$. *Compute the ciphertexts* $(c_{1,i}, c_{2,i})$ *where* $c_{1,i} = g^{s_i} \bmod N^{\xi+1}$ *and* $c_{2,i} = h^{s_i}(1+N)^{t_i} \bmod N^{\xi+1}$. *Compute the proofs* $\pi_{2,i} \leftarrow \mathsf{Prove}_{\mathcal{EPDJ}_{\Delta_{2,i}}}(crs_{2,i}, x_{2,i}, w_{2,i})$ *for* $i \in [\ell]$ *where* $x_{2,i} = (c_{1,i}, c_{2,i}, d_{1,i}, d_{2,i})$ *and* $w_{2,i} = (s_i, r, t_i)$.
- *Compute new ciphertexts* $(\bar{c}_1, \bar{c}_2)$ *and* (c_1, c_2') *where* $\bar{c}_1 = \prod_{i=1}^{\ell}(c_{1,i})^{2^{i-1}}$, $\bar{c}_2 = \prod_{i=1}^{\ell}(c_{2,i})^{2^{i-1}}$ *and* $c_2' = c_2(1+N)^{B/2}$. *Compute the proof* $\pi_{2,0} \leftarrow \mathsf{NIZK.Prove}_{\mathcal{EPDJ}_{\Delta_{2,0}}}(\mathsf{crs}_{2,0}, x_{2,0}, w_{2,0})$ *where* $x_{2,0} = ((c_1, c_2'), (\bar{c}_1, \bar{c}_2))$ *and* $w_{2,0} = (t, \bar{s}, t + B/2)$ *with* $\bar{s} = \sum_{j=1}^{\ell} s_j 2^{j-1}$.
- *Compute the proof* $\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{EDL}_{\Delta_3}}(\mathsf{crs}_3, x_3, w_3)$ *where* $x_3 = (c_1, c_2)$ *and* $w_3 = t$.
- *Output* $\pi = (D_0, d_0, c_2, \{D_i, d_{1,i}, d_{2,i}, c_{1,i}, c_{2,i}, \pi_{2,i}\}_{i\in[\ell]}, \pi_1, \pi_{2,0}, \pi_3)$.

$\mathsf{Verify}_{\mathcal{REDJ}_\Delta}(\mathsf{td}, x, \pi)$:

- *Parse* π *as* $(D_0, d_0, c_2, \{D_i, d_{1,i}, d_{2,i}, c_{1,i}, c_{2,i}, \pi_{2,i}\}_{i\in[\ell]}, \pi_1, \pi_{2,0}, \pi_3)$ *and* td *as* $(\mathsf{td}_1, \mathsf{td}_{2,0}, \mathsf{td}_3, \{\mathsf{td}_{2,i}\}_{i\in[\ell]})$
- *Compute* $\bar{c}_1 = \prod_{i=1}^{\ell}(c_{1,i})^{2^{i-1}}$, $\bar{c}_2 = \prod_{i=1}^{\ell}(c_{2,i})^{2^{i-1}}$ *and* $c_2' = c_2(1+N)^{B/2}$.
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{EQ}_{\Delta_1}}(\mathsf{td}_1, x_1, \pi_1)$ *where* $x_1 = (D_0, d_0, \{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]})$, *output* 0.
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{EPDJ}_{\Delta_{2,i}}}(\mathsf{td}_{2,i}, x_{2,i}, \pi_{2,i})$, *for all* $i \in \{0, \dots, \ell\}$, *output* 0.
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{EDL}_{\Delta_1}}(\mathsf{td}_3, x_3, \pi_3)$ *where* $x_3 = (c_1, c_2)$, *output* 0. *Else, output* 1.

Lemma 3.5.10. *The scheme presented in Construction 3.5.2 is complete.*

Proof. Let $c_1 \in \mathcal{REDJ}_\Delta$. Then, by the completeness of $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}}$, the proof π_3 is accepted. Now, the ciphertexts $(c_{1,i}, c_{2,i})$ encrypt bits t_i by the completeness of $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$. This means that the ciphertext $(\bar{c}_1, \bar{c}_2)$ encrypts $\bar{t} = \sum_{i=1}^{\ell} 2^{i-1} t_i$. Hence, $\bar{t} \in [0, B]$. By the completeness of $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}}$, (c_1, c'_2) encrypts $\bar{t} \in [0, B]$ and, thus, $(c_1, c_2 = c'_2(1+N)^{-B/2})$ encrypts $t \in [-B/2, B/2]$. We conclude that the proof is accepted as valid.

□

Lemma 3.5.11. *The scheme presented in Construction 3.5.2 has zero knowledge under the SD assumption.*

Proof. The proof follows from the fact that the schemes $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}}$ and $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}}$ are zero-knowledge (here $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$ has computational zero-knowledge under the SD assumption).

□

Lemma 3.5.12. *The scheme presented in Construction 3.5.2 is statistically simulation sound.*

Proof. The proof follows readily from the fact that the schemes $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}}$ and $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}}$ are statistically simulation sound and that the DJ scheme is linear homomorphic. That is, if $(c_{1,i}, c_{2,i})$ all encrypt bits, then $(\bar{c}_1, \bar{c}_2)$ where $\bar{c}_1 = \prod_{i=1}^{\ell} (c_{1,i})^{2^{i-1}}$ and $\bar{c}_2 = \prod_{i=1}^{\ell} (c_{2,i})^{2^{i-1}}$ is an encryption of a value smaller than $2^{\ell-1}$. □

3.6 Reusable Laconic Private Set Intersection

In this section, we present a protocol that implements ℓPSI in a black-box fashion. We then prove that the protocol guarantees security against a semi-honest receiver and against a malicious sender. The input sets are subsets of a universe $\mathcal{U}$ of exponential size.

Protocol. We now present the construction for reusable PSI.

Construction 3.6.1. *Let $\mathcal{U}$ be a universe which contains the input sets of the parties. Let $\kappa \in \mathbb{Z}$ such that $5\kappa \leq \lambda$ and $\xi \in \mathbb{N}$.*

We require the following ingredients in this construction:

1. *A PPRF* $\mathsf{PPRF} : \mathcal{K} \times \mathcal{U} \to \mathsf{Primes}(\kappa)$ *which outputs a prime number.*[19]

2. *A DV-NIZK* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZKVerify}_{\mathcal{REDJ}_\Delta})$ *for the language $\mathcal{REDJ}_\Delta$ which is defined in Section 3.5, for some $\Delta = (g_0, B, N, \xi)$.*

3. *An IND-CPA PKE scheme* $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$

[19]We remark that we use a PPRF, not because we want uniform outputs, but to implicitly define the set of primes. A similar trick was used in [BGI16].

4. *A* $(\kappa - 1, \mathsf{negl}(\lambda))$*-strong extractor* $\mathsf{Ext} : \mathcal{S} \times \mathbb{Z}_{N^{\xi+1}} \to \{0,1\}^{\lambda}$.

We assume that the receiver's set is of size M *and the sender's set is of size* m*, where* $M > m$*. The protocol is composed by the following algorithms:*

$\mathsf{GenCRS}(1^{\lambda})$:

- *Sample* $N \leftarrow\!\$\ \mathsf{RSA}(\lambda)$*, that is,* $N = PQ$ *where* P, Q *are safe prime numbers. Choose* B *such that* $N^{\xi-1}/2 \geq B > N2^{\kappa}$*.*
- *Sample a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^{\lambda})$*. Additionally, sample a PPRF key* $k \leftarrow\!\$\ \mathcal{K}$*. Set* $\Delta = (g_0, B, N, \xi)$ *where* $g_0 \leftarrow\!\$\ \mathbb{T}_N$*.*
- *Output* $\mathsf{crs} = (N, \mathsf{pk}, g_0, B, k, \Delta)$*.*

$\mathsf{R}_1(\mathsf{crs}, S_{\mathsf{R}})$:

- *Parse* $\mathsf{crs} := (N, \mathsf{pk}, g_0, B, k, \Delta)$*, and* $S_{\mathsf{R}} := \{\mathsf{id}_i\}_{i \in [M]} \subseteq \mathcal{U}$
- *Compute the prime numbers* $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}_i)$*, for all* $i \in [M]$*.*
- *Sample* $r \leftarrow\!\$\ [N/4]$ *and compute* $h = g_0^{r \prod_{i \in [M]} p_i} \bmod N^{\xi+1}$*.*
- *Run* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{REDJ}_{\Delta}}(1^{\lambda})$*.*
- *Output* $\mathsf{st} = (r, \mathsf{td}_1)$ *and* $\mathsf{psi}_1 = (h, \mathsf{crs}_1)$*.*

$\mathsf{S}(\mathsf{crs}, S_{\mathsf{S}}, \mathsf{psi}_1)$:

- *Parse* $\mathsf{crs} := (N, \mathsf{pk}, g_0, B, k, \Delta)$*,* $\mathsf{psi}_1 := (h, \mathsf{crs}_1)$ *and* $S_{\mathsf{S}} := \{\mathsf{id}'_i\}_{i \in [m]} \subseteq \mathcal{U}$*.*
- *For* $i \in [m]$ *do the following:*
  - *Sample* $\rho_i \leftarrow\!\$\ [N/4]$*. Compute the prime numbers* $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}'_i)$*.*
  - *Sample an extractor seed* $s_i \leftarrow\!\$\ \mathcal{S}$ *and compute* $R_i \leftarrow \mathsf{Ext}(s_i, h^{\rho_i} \bmod N^{\xi+1})$
  - *Compute* $f_i = g_0^{\rho_i p_i} \bmod N^{\xi+1}$ *and* $\mathsf{ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}'_i; R_i)$*.*
  - *Compute* $\pi_i \leftarrow \mathsf{NIZK.Prove}_{\mathcal{REDJ}_{\Delta}}(\mathsf{crs}_1, x_i, w_i)$ *where* $x_i = f_i$ *and* $w_i = \rho_i p_i$*.*
- *Output* $\mathsf{psi}_2 = \{f_i, \mathsf{ct}_i, s_i, \pi_i\}_{i \in [m]}$*.*

$\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$:

- *Parse* $\mathsf{st} := (r, \mathsf{td}_1)$ *and* $\mathsf{psi}_2 := \{f_i, \mathsf{ct}_i, s_i, \pi_i\}_{i \in [m]}$. *Set* $\mathcal{I} = \emptyset$
- *For all* $j \in [m]$ *do the following:*
  - *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{REDJ}_\Delta}(\mathsf{td}_1, x_j, \pi_j)$ *where* $x_j = f_j$*, abort the protocol.*
  - *If there is a* $i \in [M]$ *such that*

$$\mathsf{ct}_j = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}_i; R_i')$$

$$\textit{where } R_i' \leftarrow \mathsf{Ext}(s_j, f_j^{r_i} \bmod N^{\xi+1}) \textit{ and } r_i = r \prod_{\ell=1, \ell \neq i}^{M} p_\ell\textit{, then add the element } \mathsf{id}_i \textit{ to } \mathcal{I}.$$

- *Output* $\mathcal{I}$.

Communication cost. Here, we analyze the communication cost of the protocol as a function of the input set sizes $|S_\mathsf{S}| = m$ and $|S_\mathsf{R}| = M$ and we omit polynomial factors in the security parameter λ. The first message outputted by R_1 has size $\mathcal{O}(1)$. The second message outputted by S has size $\mathcal{O}(m)$. The overall communication cost is $\mathcal{O}(m)$, that is, it is independent of M.

Analysis. We now analyze the correctness and security of the protocol.

Theorem 3.6.1. *The protocol presented in Construction 3.6.1 is correct given that* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta}$ *is complete and* PKE *is correct.*

Proof. Let (h, crs_1) be the message sent by R_1 created using the set S_R as input.

Fix an index j such that $b_j \in S_\mathsf{S} \cap S_\mathsf{R}$. Upon receiving $(f_j, \mathsf{ct}_j, s_j, \pi_j^{\mathcal{REDJ}})$ from S_S (i.e., the part of psi_2 with respect to b_j).

Since $|\rho_j p_j| < 2^\kappa N < B$, then $1 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{REDJ}_\Delta}(\mathsf{td}_1, x_j, \pi_j)$ where $x_j = f_j$ except with negligible probability by the completeness of $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta}$.

Additionally, let $\tilde{r} = r \prod_{i: q_i \neq p_j} q_i$ where $q_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}_i)$ for $\mathsf{id}_i \in S_\mathsf{R}$ and $p_j \leftarrow \mathsf{PPRF}(k, b_j)$. Then

$$\begin{aligned} f_j^{\tilde{r}} \bmod N^{\xi+1} &= g_0^{\rho_j p_j r \prod_{i: q_i \neq p_j} q_i} \bmod N^{\xi+1} \\ &= g_0^{\rho_j r \prod_i q_i} \bmod N^{\xi+1} \\ &= h^{\rho_j} \bmod N^{\xi+1}. \end{aligned}$$

Hence, $R_j' = R_j$ and thus, $\mathsf{ct}_j = \mathsf{PKE.Enc}(\mathsf{pk}, b_j; R_j')$. Therefore, b_j is added to $\mathcal{I}$.

□

Theorem 3.6.2. *The protocol presented in Construction 3.6.1 securely UC-realizes functionality $\mathcal{F}_{\mathsf{rPSI}}$ in the $\mathcal{G}_{\mathsf{CRS}}$-hybrid model against:*

- *a semi-honest receiver given that the φ-hiding assumption hold and $\mathsf{NIZK}_{\mathcal{REDJ}_{\Delta}}$ is zero-knowledge;*
- *a malicious sender, given that the DCR assumption holds and $\mathsf{NIZK}_{\mathcal{REDJ}_{\Delta}}$ is reusable sound.*

Proof. We start by proving that the protocol is secure against semi-honest adversaries corrupting the receiver.

Lemma 3.6.3. *The protocol is secure against a semi-honest receiver.*

We first show how the simulator $\mathsf{Sim}_{\mathsf{R}}$ works. In the following, let $\mathsf{Sim}_{\mathsf{NIZK}}$ be the zero-knowledge simulator from Lemma 3.5.11 for the $\mathsf{NIZK}_{\mathcal{REDJ}_{\Delta}}$ scheme.

1. $\mathsf{Sim}_{\mathsf{R}}$ takes the input S_{R} of R and sends it to the ideal functionality $\mathcal{F}_{\mathsf{rPSI}}$.
2. **CRS generation.** To generate the CRS, Sim behaves as the honest algorithm would do.
3. The simulator creates the semi-honest receiver's view exactly as in the real protocol and keeps $\mathsf{st} = (r, \mathsf{td}_1)$ to itself.
4. Upon receiving a message $\mathsf{psi}_1 = (h, \mathsf{crs}_1)$ from R and a message $\mathcal{I}$ (of size m', that is, $|\mathcal{I}| = m'$) from the ideal functionality $\mathcal{F}_{\mathsf{rPSI}}$, the simulator does the following:
   - Sample a subset $\mathcal{X}$ of size $m - m'$ from the universe $\mathcal{U}$ and sets $S_{\mathcal{S}} = \mathcal{I} \cup \mathcal{X}$.
   - For all $i \in \mathcal{I}$, $\mathsf{Sim}_{\mathsf{R}}$ computes $(f_i, \mathsf{ct}_i, s_i, \pi_i)$ as in the real protocol.
   - For all $i \in S_{\mathsf{S}} \setminus \mathcal{I}$, $\mathsf{Sim}_{\mathsf{R}}$ simulates proofs $\pi_i \leftarrow \mathsf{Sim}_{\mathsf{NIZK}}(\mathsf{td}_1, x)$ for $x = f_i$ where $f_i \leftarrow\$ \mathbb{T}_N$. Then, it encrypts $\mathsf{ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0; R_i)$ where $R_i \leftarrow \{0,1\}^{\lambda}$.

To prove indistinguishability between the real protocol and the simulated one, we consider the following sequence of hybrids:

$\mathbf{Hyb}_0$: The is the real protocol.

$\mathbf{Hyb}_1$: This hybrid is identical to the previous one, except that, for $i \in S_{\mathsf{S}} \setminus \mathcal{I}$, $\mathsf{Sim}_{\mathsf{R}}$ simulates the proofs $\pi_i \leftarrow \mathsf{Sim}_{\mathsf{NIZK}}(\mathsf{td}_1, x)$ for $x_i = f_i$.

Claim 3.6.1. *Hybrids $\mathbf{Hyb}_0$ and $\mathbf{Hyb}_1$ are statistically indistinguishable.*

The claim above follows directly from the statistical zero-knowledge of the scheme $\mathsf{NIZK}_{\mathcal{REDJ}_{\Delta}}$.

$\mathbf{Hyb}_{2,\ell}$: This hybrid is identical to the previous one, except that the simulator samples $f_{u_\ell} \leftarrow\$ \mathbb{T}_N$ and computes

$$R_{u_\ell} \leftarrow \mathsf{Ext}\left(s, f_{u_\ell}^{rq_{u_\ell}^{-1}\prod_j^M p_j} \mod N^{\xi+1}\right)$$

where $q_{u_\ell} \leftarrow \mathsf{PPRF}(k, x_{u_\ell})$ for all $u_\ell \in \{i : x_i \in S_\mathsf{S} \setminus \mathcal{I}\}$ and $p_j \leftarrow \mathsf{PPRF}(k, y_j)$ for all $y_j \in S_\mathsf{R}$. The hybrid is defined for $\ell = 1, \ldots, m - m'$.

Claim 3.6.2. *Hybrids* $\mathbf{Hyb}_1$ *and* $\mathbf{Hyb}_{2,m-m'}$ *are indistinguishable.*

We prove that hybrids $\mathbf{Hyb}_{2,\ell-1}$ and $\mathbf{Hyb}_{2,\ell}$ are indistinguishable for $\ell = 1, \ldots, m - m'$ and where $\mathbf{Hyb}_{2,0} = \mathbf{Hyb}_1$.

First, remark that the distribution of ρ_{u_ℓ} is the uniform distribution over $[N/4]$. Hence, we can build a statistically indistinguishable sequence of hybrids $\mathbf{Hyb}_2'$ where we sample $\rho_{u_\ell} \leftarrow\$ [\varphi(N)/4]$ incurring difference only in the statistical distance.

Now, since g_0 and $g_0^{p_{u_\ell}}$ are generators of $\mathbb{T}_N$, then the distribution of $g_0^{\rho_{u_\ell} p_{u_\ell}}$ is identical to $\tilde{f}_i \leftarrow\$ \mathbb{T}_N$, for $\rho_{u_\ell} \leftarrow\$ [\varphi(N)/4]$.

For p_i sampled using PPRF (for a uniform input $x_{u_\ell} \leftarrow\$ \mathcal{U}$), we know that p_{u_ℓ} does not divide $\varphi(N)$ and $\rho_{u_\ell} \in [\varphi(N)/4]$ if $\rho_{u_\ell} \leftarrow\$ [N/4]$, except with negligible probability. We conclude that

$$\tilde{f}_{u_\ell} \mod N^{\xi+1} \approx_{\mathsf{negl}(\lambda)} g_0^{\rho_{u_\ell} p_{u_\ell}} \mod N^{\xi+1}$$

where $\tilde{f}_{u_\ell} \leftarrow\$ \mathbb{T}_N$, $g_0 \leftarrow\$ \mathbb{T}_N$, $\rho_{u_\ell} \leftarrow\$ [\varphi(N)/4]$ and $p_{u_\ell} \leftarrow\$ \mathsf{Primes}(\kappa)$.

Using a similar argument, we have that for any G

$$\left(f_{u_\ell} \mod N^{\xi+1}, f_{u_\ell}{}^{Gp_{u_\ell}^{-1}} \mod N^{\xi+1}\right) \approx_{\mathsf{negl}(\lambda)} \left(g_0^{\rho_{u_\ell} p_{u_\ell}} \mod N^{\xi+1}, g_0{}^{\rho_{u_\ell} G} \mod N^{\xi+1}\right).$$

$\mathbf{Hyb}_{3,\ell}$: This hybrid is identical to the previous one except that Sim_R computes $R_{u_\ell} \leftarrow\$ \{0,1\}^\lambda$ for all $u_\ell \in \{i : x_i \in S_\mathsf{S} \setminus \mathcal{I}\}$. The hybrid is defined for $\ell = 1, \ldots, m - m'$.

Claim 3.6.3. *Assume that* Ext *is a* $(\kappa - 1, \mathsf{negl}(\lambda))$*-strong extractor and that the* φ*-hiding assumption holds. Then hybrids* $\mathbf{Hyb}_{2,m-m'}$ *and* $\mathbf{Hyb}_{3,m-m'}$ *are indistinguishable.*

We prove that hybrids $\mathbf{Hyb}_{3,\ell-1}$ and $\mathbf{Hyb}_{3,\ell}$ are indistinguishable by constructing a reduction that contradicts Lemma 1.4.1, for $\ell = 1, \ldots, m - m'$ and where $\mathbf{Hyb}_{2,m-m'} = \mathbf{Hyb}_{3,0}$.

Suppose that there is an adversary $\mathcal{A}$ that distinguishes hybrids $\mathbf{Hyb}_{3,\ell-1}$ and $\mathbf{Hyb}_{3,\ell}$. We build an adversary $\mathcal{B}$ that breaks Lemma 1.4.1.

$\mathcal{B}$ receives as input $(N, s, q, \tilde{g})$. It behaves as the simulator in Hybrid $\mathbf{Hyb}_{3,\ell-1}$ except that it sets the modulus in the crs to be N. Additionally, it programs the PPRF such that $q \leftarrow \mathsf{PPRF}(k, x_{u_\ell})$ (this step is done while creating the PPRF key). Upon receiving a message from $\mathcal{A}$ (together with its view), it computes $G = r\prod_i^M p_i$ where $p_i \leftarrow \mathsf{PPRF}(k, x_i)$ for $x_i \in S_\mathsf{R}$. It sends G to the challenger and receives $\tilde{z}$. This value $\tilde{z}$ is either equal to $\mathsf{Ext}(s, \tilde{g}^{G/q} \mod N^{\xi+1})$, if $\gamma = 0$, or it is uniformly chosen, if $\gamma = 1$, where γ is the challenge bit. Now $\mathcal{B}$ sets $f_{u_\ell} = \tilde{g}$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x_{u_\ell}; R_{u_\ell})$ and sends psi_2 as in Hybrid $\mathcal{H}_{3,\ell-1}$ except that the u_ℓ-th coordinate is $(f_{u_\ell}, \mathsf{ct}_{u_\ell}, s_{u_\ell}, \pi_{u_\ell})$. The adversary outputs a bit b and $\mathcal{B}$ sets b as its guess. It is easy to see that if $\gamma = 0$, then $\mathcal{B}$'s message is indistinguishable from the message of hybrid $\mathbf{Hyb}_{3,\ell-1}$ and if $\gamma = 1$, then it is indistinguishable from the message sent in hybrid $\mathbf{Hyb}_{3,\ell}$.

$\mathbf{Hyb}_{4,\ell}$: This hybrid is identical to the previous one except that Sim_R encrypts $\mathsf{ct}_{u\ell} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0; R_{u_\ell})$ for all for all $u_\ell \in \{i : x_i \in S_\mathsf{S} \backslash \mathcal{I}\}$. The hybrid is defined for $\ell = 1, \ldots, m-m'$. Hybrid $\mathbf{Hyb}_{4,m-m'}$ is identical to the simulation.

Claim 3.6.4. *Assume that* PKE *is an IND-CPA PKE. Then hybrids* $\mathbf{Hyb}_{3,m-m'}$ *and* $\mathbf{Hyb}_{4,m-m'}$ *are indistinguishable.*

The claim follows directly from the IND-CPA property of the underlying PKE. That is, given an adversary $\mathcal{A}$ that distinguishes both hybrids, we can easily build an adversary $\mathcal{B}$ against the IND-CPA property of PKE. This adversary $\mathcal{B}$ simply chooses as messages $m_0 = x_{u_\ell}$ (where $x_{u_\ell} \in S_\mathsf{S} \setminus \mathcal{I}$) and $m_1 = 0$. It outputs whatever $\mathcal{A}$ outputs.

Lemma 3.6.4. *The protocol is secure against malicious senders.*

We first show how the simulator Sim_S extracts the sender's input:

1. **CRS generation.** Sim_S generates the crs following the algorithm GenCRS, except that it sets $g_0 = g_0'(1+N)$ for $g_0' \leftarrow\$ \mathbb{T}_N$. It keeps $\varphi(N)$ to itself (which can be computed using the prime numbers P, Q) and the secret key sk corresponding to pk. It outputs $\mathsf{crs} = (\mathsf{pk}, g_0, B, k, \Delta)$
2. Sim_S samples $h \leftarrow\$ \mathbb{T}_N$ and computes $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{REDJ}_\Delta}(1^\lambda)$. It sends $\mathsf{psi}_1 = (h, \mathsf{crs}_1)$ to the malicious sender.
3. Whenever Sim_S receives a message $\mathsf{psi}_2 = \{f_i, \mathsf{ct}, s_i, \pi_i\}_{i\in[m]}$ from the sender, the simulator initially sets S_S and does the following for all $i \in [m]$:
    - It checks if $1 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{REDJ}_\Delta}(\mathsf{td}_1, x_j, \pi_j)$ where $x_j = f_j$, and aborts otherwise.
    - It computes $\mathsf{id}_i' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}_i)$ and $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}_i')$. Additionally, it extracts ζ_i by recovering ζ_i' from $(1+N)^{\zeta_i'} = f_i^{\varphi(N)}$ and computing $\zeta = \zeta'/\varphi(N)$ over the integers. It computes $\rho_i' = \zeta_i/p_i$ over the integers. If $\mathsf{ct}_i = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}_i'; R_i)$ where $R_i = \mathsf{Ext}(s_i, h^{\rho_i'} \bmod N^{\xi+1})$, then it adds id_i' to S_S.
4. It sends S_S to $\mathcal{F}_\mathsf{PSI}$ and halts.

We now show that the simulation is indistinguishable from the real protocol via the following sequence of hybrids.

$\mathbf{Hyb}_0$: This hybrid is the real protocol.

Hyb$_1$: This hybrid is identical to the previous one except that the simulator computes the first message (sent by the receiver) as $h \leftarrow\!\!\$\ \mathbb{T}_N$.

Claim 3.6.5. *Hybrids* **Hyb$_0$** *and* **Hyb$_1$** *are statistically indistinguishable.*

Since g_0 is a generator of $\mathbb{T}_N$, the distributions of g^x and $h \leftarrow\!\!\$\ \mathbb{T}_N$ are identical. It follows that the hybrids are indistinguishable.

Hyb$_2$: This hybrid is identical to the previous one, except that $g_0 = g_0'(1+N)$ for $g_0' \leftarrow\!\!\$\ \mathbb{T}_N$ (instead of choosing $g_0 \leftarrow\!\!\$\ \mathbb{T}_N$). Additionally, Sim_S keeps $(\varphi(N), \mathsf{sk})$ while creating crs.

Claim 3.6.6. *Assume that the DCR assumption holds. Then hybrids* **Hyb$_1$** *and* **Hyb$_2$** *are indistinguishable.*

The claim follows directly from Corollary 1.4.3.

Hyb$_3$: This hybrid is identical to the previous one except that the simulator, instead of checking if there is an index i for which

$$\mathsf{ct}_j = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}_i; R_i')$$

where $R_i' = \mathsf{Ext}(s_j, f_j^{r_i})$ and $r_i = r \prod_{\ell=1,\ell\neq i}^{M} p_\ell$ (as in the real protocol), it does the checks as in the simulation. That is, it computes $\mathsf{id}_i' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}_i)$ and $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}_i')$. Additionally, it extracts ζ_i by recovering ζ_i' from $(1+N)^{\zeta_i'} = f_i^{\varphi(N)}$ and computing $\zeta = \zeta'/\varphi(N)$. It computes $\rho_i' = \zeta_i/p_i$ over the integers. Then, it checks if $\mathsf{ct}_i = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}_i'; R_i)$ where $R_i = \mathsf{Ext}(s_i, h^{\rho_i'})$.

Claim 3.6.7. *Hybrids* **Hyb$_2$** *and* **Hyb$_3$** *are indistinguishable given that* PKE *is correct and* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta}$ *is simulation sound.*

By the simulation soundness of $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta}$, $\zeta_i < N^{\xi-1}/2$. Hence, $\zeta_i' < N^{\xi}/2$ and thus $\zeta_i' \bmod N^{\xi}$ is equal to ζ_i' as an integer. Computing $\zeta = \zeta_i'/\varphi(N)$ yields $\rho_i p_i$ over $\mathbb{Z}$. Thus $\rho_i = \zeta_i/p_i$ over $\mathbb{Z}$.

Thus, performing the checks in this hybrid has the same outcome as in the real protocol.

□

Setting the parameters. The value B is such that $N^{\xi-1}/2 \geq B > N2^{\kappa}$ for $5\kappa \leq \lambda$. Then, it is enough to set $\xi = 3$, so that we can find a B that fulfills the condition.

Achieving statistical security against the sender. The protocol presented in Construction 3.6.1 achieves computational security against a malicious sender given that the DCR assumption holds (recall that $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta}$ achieves statistical reusable soundness).

The only place where we use the DCR assumption in the proof of security against a malicious sender is when we replace $g_0 \leftarrow\$ \mathbb{T}_N$ by $g_0 = g_0'(1+N)$. Hence, consider the following modification of the protocol presented in Construction 3.6.1: In GenCRS, the element g_0 is chosen as $g_0'(1+N)$ for g_0'. This simple modification of the protocol yields a new one which is *statistically secure against a malicious sender*. On the other hand, security against a semi-honest receiver now relies on the hardness of φ-hiding (as before) *and* the DCR assumption.

3.7 Labeled Laconic PSI and Laconic OT

In this section, we show how we can extend the techniques developed in Section 3.6 to construct LPSI to obtain new constructions of labelled LPSI and LOT. Both constructions are reusable and secure against malicious senders.

3.7.1 Reusable Labeled Laconic PSI Secure Against a Malicious Sender

Reusable Labeled PSI functionality. The functionality $\mathcal{F}_{\mathsf{rLPSI}}$ is parametrized by a universe $\mathcal{U}$ and by a universe of labels $\mathcal{L}$ and works as follows:

- **Setup phase.** R sends $(\mathsf{sid}, S_{\mathsf{R}})$ to $\mathcal{F}_{\mathsf{rLPSI}}$ where $S_{\mathsf{R}} \subseteq \mathcal{U}$. It ignores future messages from R with the same sid.
- **Send phase.** S sends $(\mathsf{sid}, i, S_{\mathsf{S},\mathsf{lab}} \subseteq \mathcal{U} \times \mathcal{L})$ from S to $\mathcal{F}_{\mathsf{rLPSI}}$. $\mathcal{F}_{\mathsf{rLPSI}}$ sends $(\mathsf{sid}, i, S_{\mathsf{R}\cap S_{\mathsf{S}},\mathsf{lab}})$ to R, where $S_{\mathsf{R}\cap S_{\mathsf{S}},\mathsf{lab}} = \{(y,\ell) \in S_{\mathsf{S},\mathsf{lab}} : y \in S_{\mathsf{R}}\}$. It ignores future messages from S with the same sid and $i \in \mathbb{N}$.

Protocol. We now present the construction for labeled reusable PSI.

Construction 3.7.1. *Let $\mathcal{U}$ be a universe which contains the input sets of the parties. Let $\kappa \in \mathbb{Z}$ such that $5\kappa \leq \lambda$. Let*

- $\mathsf{PRF} : \mathcal{K} \times \mathcal{U} \to \mathsf{Primes}(\kappa)$ *be a PRF which outputs prime numbers*
- $\mathcal{REDJ}_\Delta$ *be the language defined in Section 3.5 and* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZKVerify}_{\mathcal{REDJ}_\Delta})$ *be a DV-NIZK for the language $\mathcal{REDJ}_\Delta$, for some $\Delta = ((g_0, g_1), B, N, \xi)$.*
- $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *be an IND-CPA PKE scheme*
- $\mathsf{Ext} : \mathcal{S} \times \mathbb{Z}_{N^{\xi+1}} \to \{0,1\}^{2\lambda}$ *be a $(\kappa - 1, \mathsf{negl}(\lambda))$-strong extractor.*

We assume that the receiver's set is of size M and the sender's set is of size m, where $M > m$. The protocol is composed of the following algorithms:

GenCRS : *This algorithm is identical to the one described in Construction 3.6.1.*

$\mathsf{R}_1(\mathsf{crs}, S_\mathsf{R})$: *This algorithm is identical to the one described in Construction 3.6.1.*

$\mathsf{S}(\mathsf{crs}, S_\mathsf{S}, \mathsf{psi}_1)$: *This algorithm is identical to the one described in Construction 3.6.1, except that* $(R_i||T_i) \leftarrow \mathsf{Ext}(s, h^{\rho_i} \bmod N^{\xi+1})$*. The string* R_i *is used to encrypt the set element (as in Construction 3.6.1) Additionally, compute* $\bar{\mathsf{ct}}_i = T_i \oplus \mathsf{lab}_i$*, where* lab_i *is the corresponding label.*

$\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$: *This algorithm is identical to the one described in Construction 3.6.1, except that whenever* $a_i \in \mathcal{I}$*, compute* $\mathsf{lab}_i = T_i \oplus \bar{\mathsf{ct}}_i$*. Output* $(\mathcal{I}, \{\mathsf{lab}_i\}_{i\in\mathcal{I}})$*.*

Analysis. We state the theorems that guarantee the required properties for our scheme. We omit the proofs since they are identical to the proofs of Theorems 3.6.1 and 3.6.2

Theorem 3.7.1. *The protocol presented in Construction 3.7.1 is correct.*

Theorem 3.7.2. *The protocol presented in Construction 3.7.1 securely UC-realizes functionality* $\mathcal{F}_{\mathsf{rLPSI}}$ *in the* $\mathcal{G}_{\mathsf{CRS}}$*-hybrid model against:*

- *a semi-honest receiver given that the* φ*-hiding and the DCR assumptions hold;*
- *a malicious sender, where security holds statistically.*

3.7.2 Laconic Oblivious Transfer with Malicious Sender Security

In this section, we present a new laconic oblivious transfer (LOT) scheme which is secure against the malicious sender. Besides, it only needs a small CRS and succinct messages for both rounds (as in [GVW20]).

Laconic oblivious transfer ideal functionality. Let $\Gamma = \Gamma(\lambda) \in \mathbb{N}$. The functionality $\mathcal{F}_{\ell\mathsf{OT}}$ works as follows: It receives a database $D \in \{0,1\}^\Gamma$ from R. Upon receiving a message $(i \in \mathbb{N}, m_0, m_1, L \in [\Gamma])$ from the sender S, $\mathcal{F}_{\ell\mathsf{OT}}$ sends (i, m_{D_L}) to R and ignores future messages with the same i from S.

Protocol. We now present the construction for sender-malicious LOT.

Construction 3.7.2. *Let* $\Gamma = \Gamma(\lambda)$ *be a polynomial in* λ*. Let Let* $\kappa \in \mathbb{Z}$ *such that* $5\kappa \leq \lambda$*. Let*

- $\mathsf{PRF} : \mathcal{K} \times \mathcal{U} \to \mathsf{Primes}(\kappa)$ *be a PRF which outputs prime numbers*
- $\mathcal{REDJ}_\Delta$ *be the language defined in Section 3.5 and* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZKVerify}_{\mathcal{REDJ}_\Delta})$ *be a DV-NIZK for the language* $\mathcal{REDJ}_\Delta$*, for some* $\Delta = ((g_0, g_1), B, N, \xi)$*.*
- $\mathsf{Ext} : \mathcal{S} \times \mathbb{Z}_{N^{\xi+1}} \to \{0,1\}^{2\lambda}$ *be a* $(\kappa - 1, \mathsf{negl}(\lambda))$*-strong extractor.*

$\mathsf{GenCRS}(1^\lambda)$: *This algorithm is identical to the one described in Construction 3.6.1, except that it does not create a public key* pk. *It outputs* $\mathsf{crs} = (N, (g_0, g_1), B, k, \Delta)$ *where* $\Delta = ((g_0, g_1), B, N, \xi)$.

$\mathsf{Hash}(\mathsf{crs}, D \in \{0,1\}^\Gamma)$: *It computes* $h = g_0^{r\prod_{i=1}^{\Gamma} e_{i,D_i}} \mod N^{\xi+1}$, *where* $r \leftarrow\!\!\$ \ [N/4]$ *and* $e_{i,b} \leftarrow \mathsf{PPRF}(k, 2i+b)$ *for* $i \in [\Gamma]$ *and* $b \in \{0,1\}$, *and computes* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{ERDJ}_\Delta}(1^\lambda)$. *It outputs* $\mathsf{lot}_1 = (h, \mathsf{crs}_1)$.

$\mathsf{Send}(\mathsf{crs}, \mathsf{lot}_1, m_0, m_1, L)$: *It computes* $f_j = g_0^{\rho_j e_{L,j}}$, $F_j = g_1^{\rho_j e_{L,j}}(1+N))^{\rho_j e_{L,j}}$ *for* $j \in \{0,1\}$ *where* $\rho_j \leftarrow\!\!\$ \ [N/4]$ *and* $e_{L,j} \leftarrow \mathsf{PPRF}(k, 2L+j)$, *computes* $\mathsf{ct}_j = k_j \oplus m_j$, *where* $k_j \leftarrow \mathsf{Ext}(s_j, h^{\rho_j})$, *computes* $\pi_j \leftarrow \mathsf{NIZK.Prove}_{\mathcal{ERDJ}_\Delta}(\mathsf{crs}, x_j, w_j)$ *where* $x_j = (f_j, F_j)$ *and* $w_j = (\rho_j e_{L,j})$ *It outputs* $\mathsf{lot}_2 = (\{f_j, F_j, \mathsf{ct}_j, \pi_j, s_j\}_{j\in\{0,1\}}, L)$.

$\mathsf{Receive}(\mathsf{crs}, \mathsf{lot}_2, \mathsf{st})$: *It aborts if* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{ERDJ}_\Delta}(\mathsf{td}, x_j, \pi_j)$ *where* $x_j = (f_j, F_j)$. *It computes* $k_{D_L} \leftarrow \mathsf{Ext}(s_{D_L}, f_{D_L}^{r\prod_{i\neq L} e_{i,D_i}} \mod N^{\xi+1})$, *and outputs* $m_{D_L} = \mathsf{ct}_{D_L} \oplus k_{D_L}$.

Analysis. We state the theorems that guarantee the required properties for our scheme.

Theorem 3.7.3. *The protocol presented in Construction 3.7.2 is correct.*

The proof of correctness essentially follows the same lines as the proof of Theorem 3.6.1.

Theorem 3.7.4. *The protocol presented in Construction 3.7.2 securely UC-realizes functionality* $\mathcal{F}_{\ell\mathsf{OT}}$ *in the* $\mathcal{G}_{\mathsf{CRS}}$*-hybrid model against:*

- *a semi-honest receiver given that the* φ*-hiding and the DCR assumptions hold;*
- *a malicious sender, where security holds statistically.*

Proof. The proof of security against a semi-honest receiver is identical to the proof of Lemma 3.6.3.

We now sketch how to prove security against a malicious sender. The simulator works analogously to the simulator of Lemma 3.6.4, except that, in this case, the simulator knows the prime $e_{L,i}$ for both $i \in \{0,1\}$. Thus, the re-encryption step is not needed anymore since the simulator can easily extract ρ_i, for $i \in \{0,1\}$ by decrypting (f_i, F_i) using $\varphi(N)$ (which is well-formed and encrypting a value smaller than N^2 by the soundness of $\mathsf{NIZK}_{\mathcal{ERDJ}_\Delta}$) to recover a value ζ_i'. From this value, it can compute $\rho_i = \zeta_i'/(e_{L,i}\varphi(N))$. After recoreving ρ_i, it can compute the keys k_i and extract the messages m_i.

Indistinguishability between the simulated version and the real protocol follows the same blueprint as the proof of Lemma 3.6.4. □

3.8 Self-Detecting Encryption

In this section, we define self-detecting encryption and show how to build it from laconic PSI. We first give a semi-honest definition and will present the malicious definition in the full paper [ABD+21b].

Definition 3.8.1. A Self-Detecting Encryption (SDE) scheme is a tuple of (randomized) algorithms $\mathsf{SDE} = (\mathsf{Prm}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Detect})$ such that:

- $\mathsf{Prm}(1^\lambda)$: Takes as input a security parameter 1^λ, and outputs a public parameter p.
- $\mathsf{Gen}(\mathsf{p})$: Takes as input a public parameter p, and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Hash}(\mathsf{p}, \mathsf{DB})$: Takes as input a public parameter p and a database DB, and outputs a hash value h and a private state st. We require $|\mathsf{h}| \leq \mathsf{poly}(\lambda)$, for a fixed polynomial poly.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{msg})$: Takes as input a public key pk, a hash value h, and a message msg, and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Takes as input a secret key sk and a ciphertext ct, and outputs a message msg or $\perp$.
- $\mathsf{Detect}(\mathsf{st}, \mathsf{ct})$: Takes as input a private state st and a ciphertext ct, and outputs a message msg or $\perp$.

We require the following properties:

- **Correctness.** For any message msg, letting $\mathsf{p} \leftarrow_\$ \mathsf{Prm}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(\mathsf{p})$: $\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})) \neq \mathsf{msg}] \leq \mathsf{negl}(\lambda)$.
- **Detection.** For any $\mathsf{p} \in \mathsf{Prm}$, any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$, any database of strings DB, and any message msg, letting $(\mathsf{h}, \mathsf{st}) \leftarrow_\$ \mathsf{Hash}(\mathsf{p}, \mathsf{DB})$ and $\mathsf{ct} \leftarrow_\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{msg})$, if $\mathsf{msg} \in \mathsf{DB}$ then $\mathsf{Detect}(\mathsf{st}, \mathsf{ct}) = \mathsf{msg}$.
- **Efficiency.** The size of h and running time of Enc are independent of the database size. There exists a polynomial poly s.t. for all $n := n(\lambda)$, any $\mathsf{DB} \in \{0,1\}^n$, letting $h \leftarrow_\$ \mathsf{Hash}(\mathsf{p}, \mathsf{DB})$ and p, pk be as above, then $|h| \leq \mathsf{poly}(\lambda)$ and also the running time of $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{msg})$ is upper bounded by $\mathsf{poly}(|\mathsf{msg}|, \lambda)$.
- **Database Hiding.** For any two databases $(\mathsf{DB}_0, \mathsf{DB}_1)$ of equal size, if $(\mathsf{h}_0, *) \leftarrow_\$ \mathsf{Hash}(\mathsf{p}, \mathsf{DB}_0)$ and $(\mathsf{h}_1, *) \leftarrow_\$ \mathsf{Hash}(\mathsf{p}, \mathsf{DB}_1)$ then h_0 and h_1 are indistinguishable where $\mathsf{p} \leftarrow_\$ \mathsf{Gen}(1^\lambda)$.
- **Semantic Security.** For any database of strings DB and any two messages $(\mathsf{msg}_0, \mathsf{msg}_1)$: $(\mathsf{pk}, \mathsf{h}, \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_0)) \overset{c}{\equiv} (\mathsf{pk}, \mathsf{h}, \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_1))$, where all the variables are sampled as above.

- **Security Against the Authority.** For any two messages $(\mathsf{msg}_0, \mathsf{msg}_1)$, if $\mathsf{msg}_0 \notin \mathsf{DB}$ and $\mathsf{msg}_1 \notin \mathsf{DB}$ then

$$\big(\mathsf{pk}, (\mathsf{h}, \mathsf{st}), \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_0)\big) \overset{c}{\equiv} \big(\mathsf{pk}, (\mathsf{h}, \mathsf{st}), \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_1)\big),$$

  where $\mathsf{p} \leftarrow_\$ \mathsf{Prm}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(\mathsf{p})$, and $(\mathsf{h}, \mathsf{st}) \leftarrow_\$ \mathsf{Hash}(\mathsf{p}, \mathsf{DB})$.

We now show how to realize self-detecting encryption from semi-honest laconic PSI. Informally, the SDE hash is the receiver's first-round laconic PSI message, and the encryption of a message m consists of a PKE encryption of m as well as a second-round PSI message based on m.

Construction 3.8.1. *Let* $\mathsf{PKE} = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ *be a CPA-secure PKE scheme*[20] *and* $\mathsf{LPSI} = (\mathsf{GenCRS}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ *a laconic PSI.*

- $\mathsf{Prm}(1^\lambda)$*: Sample* $\mathsf{crs} \leftarrow_\$ \mathsf{LPSI.GenCRS}(1^\lambda)$*, and let* $\mathsf{p} := \mathsf{crs}$.
- $\mathsf{Gen}(\mathsf{p})$*: Run* $\mathsf{PKE.Gen}'(1^\lambda)$ *to generate a pair of keys* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Hash}(\mathsf{p}, \mathsf{DB})$*: Let* h *be the output of the receiver on* DB *and* p*, i.e.,* $\mathsf{h} \leftarrow_\$ \mathsf{LPSI.R}_1(\mathsf{p}, \mathsf{DB})$*. In addition, let* st *be the private state of the receiver.*
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{msg})$*: Output* $(\mathsf{ct}_1, \mathsf{ct}_2)$*, where* $\mathsf{ct}_1 \leftarrow_\$ \mathsf{PKE.Enc}'(\mathsf{pk}, \mathsf{msg})$ *and* $\mathsf{ct}_2 \leftarrow_\$ \mathsf{LPSI.S}(\mathsf{p}, \{\mathsf{msg}\}, \mathsf{h})$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$*: Output* $\mathsf{PKE.Dec}'(\mathsf{sk}, \mathsf{ct}_1)$.
- $\mathsf{Detect}(\mathsf{st}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$*: Output* $\mathsf{R}_2(\mathsf{st}, \mathsf{ct}_2)$.

Correctness and efficiency follow immediately.

- **Statistical** database hiding follows from PSI-receiver statistical security.
- Semantic security and security against the authority property of the scheme follows from the CPA security of PKE scheme Π and the sender's security. Observe that if $\mathsf{msg} \notin \mathsf{DB}$ then both ct_1 and ct_2 computationally hide the message even in the presence of the private state st of PSI. Specifically, one can argue that ct_1 computationally hides msg because of the CPA security of PKE scheme Π, and ct_2 computationally hides msg because of the sender's security of laconic PSI. The arguments above can be made formal via a routine hybrid argument, and we omit the details.

[20] We proceed with an independent PKE scheme for the sake of simplicity.

3.8.1 Maliciously Secure Self-Detecting Encryption

Next, we provide a definition of self-detecting encryption in the malicious setting. In this setting, the algorithm Prm provides a trapdoor which allows a server to ensure that the ciphertexts sent on the channel can be verified while ensuring the privacy of users. We remark that the trapdoor is only known for the server (and is not included in the user's secret key).[21] Clearly, as in the semi-honest setting, the authority would only be able to detect illegal content by looking at the ciphertexts communicated through the channel, and no information will be leaked about normal/legal messages.

Specifically, a maliciously secure self-detecting encryption is a tuple of seven (randomized) algorithms $\mathsf{SDE} = (\mathsf{Prm}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Detect}, \mathsf{Verify})$ such that Prm outputs a trapdoor td (along with p) such that the verification algorithm Verify checks well-formedness of ciphertext using td as follows:

- $\mathsf{Verify}(\mathsf{td}, \mathsf{ct})$: Takes a trapdoor td and a ciphertext ct and it outputs 1 or 0.

The other five algorithms, namely $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Detect})$, have the same functionality as in the semi-honest setting. We require that SDE should satisfy all the properties of a semi-honest encryption scheme (correctness, efficiency, database hiding, semantic security, and security against the authority), along with the following well-formedness property: for any PPT adversary $\mathcal{A}$ and database DB, if $(\mathsf{p}, \mathsf{td}) \leftarrow_\$ \mathsf{Prm}(1^\lambda)$, $(\mathsf{pk}, *) \leftarrow_\$ \mathsf{Gen}(\mathsf{p})$, $(\mathsf{h}, *) \leftarrow_\$ \mathsf{Hash}(\mathsf{p}, \mathsf{DB})$ then the following holds for any adversarially generated ciphertext $\mathsf{ct} \leftarrow_\$ \mathcal{A}^{\mathsf{Verify}(\mathsf{td},.)}(\mathsf{p}, \mathsf{pk}, \mathsf{h})$ with overwhelming probability (where $\mathcal{A}$ has oracle access to the verification algorithm):

- If $\mathsf{Verify}(\mathsf{td}, \mathsf{ct}) = 1$ and $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \in \mathsf{DB}$ then $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{Detect}(\mathsf{st}, \mathsf{ct})$.
- If $\mathsf{Verify}(\mathsf{td}, \mathsf{ct}) = 1$ and $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \notin \mathsf{DB}$ then $\mathsf{Detect}(\mathsf{st}, \mathsf{ct}) = \perp$.

Given a maliciously secure laconic PSI and a DV-NIZK for a specific language, one can construct a maliciously secure SDE following the same blueprint that we provided in the semi-honest setting.

Construction 3.8.2. *Let* $\mathsf{PKE} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *be a CPA-secure public-key encryption scheme, and let* $\mathsf{NIZK} = (\mathsf{NIZK.GenCRS}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ *be a DV-NIZK for "message-equality" language (described below).*

- $\mathsf{Prm}(1^\lambda)$*: Sample* $(\mathsf{crs}_N, \mathsf{td}) \leftarrow_\$ \mathsf{NIZK.GenCRS}(1^\lambda)$ *and* $\mathsf{crs}_L \leftarrow_\$ \mathsf{LPSI.GenCRS}(1^\lambda)$*, and let* $\mathsf{p} = (\mathsf{crs}_N, \mathsf{crs}_L)$.
- $\mathsf{Gen}(\mathsf{p})$*: Sample a pair of keys* $(\mathsf{pk}', \mathsf{sk}') \leftarrow_\$ \mathsf{PKE.Gen}'(1^\lambda)$*. Set* $\mathsf{pk} = (\mathsf{pk}', \mathsf{crs}_N, \mathsf{crs}_L)$ *and* $\mathsf{sk} = \mathsf{sk}'$.
- $\mathsf{Hash}(\mathsf{p}, \mathsf{DB})$*: Parse* $\mathsf{p} = (\mathsf{crs}_N, \mathsf{crs}_L)$*. Output* $(\mathsf{h}, \mathsf{st}) \leftarrow_\$ \mathsf{LPSI.R}_1(\mathsf{crs}_L, \mathsf{DB})$.

[21] Notice that in the malicious setting, there are three entities (user, server, and the authority) with their own secret key/state.

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{msg})$*: Parse* $\mathsf{pk} = (\mathsf{pk}', \mathsf{crs}_L, \mathsf{crs})$. *Let* $\mathsf{ct}_1 \leftarrow_\$ \mathsf{PKE.Enc}'(\mathsf{pk}', \mathsf{msg})$ *and* $\mathsf{ct}_2 \leftarrow_\$ \mathsf{LPSI.S}(\mathsf{crs}_L, \{\mathsf{msg}\}, \mathsf{h})$.

  Compute proof for the statement that the messages underlying ct_1 *and* ct_2 *are equal. Specifically, consider the following language* $\mathcal{L}$ *(parameterized by* Δ*):*

  $$\mathcal{L}_\Delta = \{(\mathsf{ct}_1, \mathsf{ct}_2) : \exists(\mathsf{msg}, r, r') \textit{ s.t. } \mathsf{ct}_1 = \mathsf{PKE.Enc}'(\mathsf{pk}', \mathsf{msg}; r') \wedge \mathsf{ct}_2 = \mathsf{LPSI.S}(\mathsf{crs}_L, \{\mathsf{msg}\}, \mathsf{h}; r)\},$$

  where $\Delta = (\mathsf{pk}', \mathsf{crs}_L, \mathsf{h})$. *In addition, r and r′ are the random coins used by* $\mathsf{PKE.Enc}'$ *and* $\mathsf{LPSI.S}$*, respectively. Generate a proof* $\pi \leftarrow_\$ \mathsf{NIZK.Verify}(\mathsf{crs}_N, \mathsf{ct}_1, \mathsf{ct}_2)$*, and set* $\mathsf{ct}_3 := \pi$. *Finally, publish* $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$ *as the ciphertext.*

- $\mathsf{Verify}(\mathsf{td}, \mathsf{ct})$ *: Run* $\mathsf{NIZK.Verify}$ *on* td *and* ct*, and output the resulting bit.*

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3))$*: Output* $\mathsf{PKE.Dec}'(\mathsf{sk}', \mathsf{ct}_1)$.

- $\mathsf{Detect}(\mathsf{st}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3))$*: Output* $\mathsf{LPSI.R}_2(\mathsf{st}, \mathsf{ct}_2)$.

Correctness, efficiency, database hiding, semantic security, and security against the authority of the scheme can be argued in a similar fashion to the semi-honest setting. The additional requirement, namely the well-formedness property of the scheme essentially follows from the security of DV-NIZK. Observe that for a maliciously generated ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$, the messages hidden by ct_1 and ct_2 are not equal, and hence the ciphertext ct will be rejected by the verification algorithm of DV-NIZK. We leave a black-box construction of DV-NIZK (for the message-equality language above) from concrete cryptographic assumptions to future work.

4 Rate-1 Oblivious Transfer

In this chapter, we start by addressing the third problem related to privacy-preserving computation, which involves the communication bandwidth of a fundamental cryptographic primitive known as oblivious transfer (OT) which we will recall in Section 4.1.

In particular, we show that it is possible to perform n independent copies of 1-out-of-2 oblivious transfer in two messages, where the communication complexity of the receiver and sender (each) is $n(1 + o(1))$ for sufficiently large n. Note that this matches the information-theoretic lower bound. Prior to this thesis, this was only achievable by using the heavy machinery of rate-1 fully homomorphic encryption (Rate-1 FHE[BDGM19]).

To achieve rate-1 both on the receiver's and sender's end, we use the LPN assumption, with slightly sub-constant noise rate $1/m^{\varepsilon}$ for any $\varepsilon > 0$ together with either the DDH, QR or LWE assumptions[1]. In terms of efficiency, our protocols only rely on linear homomorphism, as opposed to the FHE-based solution which inherently requires an expensive "bootstrapping" operation. We believe that in terms of efficiency, we compare favourably to existing batch-OT protocols while achieving superior communication complexity. We show similar results for Oblivious Linear Evaluation (OLE).

For our DDH-based solution, we develop a new technique that may be of independent interest. We show that it is possible to "emulate" the binary group $\mathbb{Z}_2$ (or any other small-order group) inside a prime-order group $\mathbb{Z}_p$ *in a function-private manner*. That is, $\mathbb{Z}_2$ operations are mapped to $\mathbb{Z}_p$ operations such that the outcome of the latter does not reveal additional information beyond the $\mathbb{Z}_2$ outcome. Our encoding technique uses the discrete Gaussian distribution, which to our knowledge was not done before in the context of DDH.

[1] Similar as before, these hardness assumptions are recalled in Chapter 1.

4.1 Overview

Oblivious Transfer (OT) [Rab05, EGL82] is one of the most basic cryptographic primitives. In the simple 1-out-of-2 OT, a receiver holds a bit $b \in \{0,1\}$ and a sender holds two bits x_0, x_1. At the end of the protocol, the receiver should learn x_b, but nothing about x_{1-b}, and the sender should learn nothing about the value of b. In most applications, one OT is not enough and one is required to perform many OT operations in parallel. We let n denote the number of parallel executions. Various techniques have been developed to address this task of *batch-OT* [IKNP03, BCG$^+$19b, BCG$^+$19a].

For the most part, they involve a preprocessing "offline" phase where the parties generate random OT correlations.[2] Given such correlations, executing the OT protocol in the so-called "online phase" is computationally very simple. This approach is very useful for purposes of computational efficiency since the offline phase can be carried out even before the actual inputs of the computation are known. However, in terms of communication complexity, there is an inherent cost, even just in the online phase, of n receiver bits and $2n$ sender bits. In contrast, the insecure implementation only requires n bits to be sent from each party in a two-message protocol: the receiver sends its input, and the sender returns all of the appropriate x_b values. As always in cryptography, we wish to understand what is the "cost of privacy", namely can we approach the information-theoretic minimum without losing privacy. Note that we can only hope to achieve this for a sufficiently large n, due to the security parameter overhead.[3]

In prior work, Döttling et al. [DGI$^+$19] showed that if the same receiver bit is used for multiple OT instances, then the sender's response can be compressed to $n(1 + o(1))$, achieving an optimal amortized rate. This was shown under a variety of computational assumptions: Decisional Diffie-Hellman (DDH), Quadratic Residuosity (QR), or Learning with Errors (LWE). It was also shown by Brakerski et al. [BDGM19] and by Gentry and Halevi [GH19] that fully homomorphic encryption (FHE) can achieve optimal communication complexity, which in particular implies that under the LWE assumption, optimal rate batch-OT is achievable. However, the FHE-based protocol inherently requires the use of a computationally exorbitant "bootstrapping" mechanism in order to compress the receiver's message.

[2] That is, a protocol in which the receiver obtains b, x_b and the sender obtains x_0, x_1, where b, x_0, x_1 are all (pseudo-)randomly sampled.

[3] In more detail, since 2-message OT implies a public-key encryption scheme, the messages must have a length that relates to the security parameter of the underlying computation assumption. This is the case even for single-bit OT.

4.1.1 Our Contribution

We show that optimal-rate[4] batch-OT can be achieved from various computational assumptions, and without giving up on computational efficiency. In particular, we require the LPN assumption with a small-inverse-polynomial noise[5], in addition to one of the assumptions DDH, QR or LWE. In terms of computational cost, our protocol does not require heavy operations such as bootstrapping and relies on linear homomorphism only. We believe that in terms of overall cost, it compares favourably even with random-OT-based methods. All of our results are in the semi-honest (honest-but-curious) setting.

We further extend our results to the task of Oblivious Linear Evaluation (OLE) [IPS09, CDI+19, GNN17, BDM22], where the sender holds a linear function over a ring and the receiver holds an input for the function, and we wish for the receiver to learn the output on its input and nothing more, and the sender learns nothing as usual. OLE has been shown to be useful in various settings [GMW19, CDI+19].

Our techniques rely primarily on linear homomorphism, namely on the ability to evaluate linear functions on encrypted data (see Section 4.2 below). We require a linearly homomorphic scheme over $\mathbb{Z}_2$ (more generally $\mathbb{Z}_q$ for OLE) where the evaluation is *function-private*. Namely, the output ciphertext should not reveal any information about the linear function that was evaluated. This was not known to be achievable from DDH prior to this thesis, and we introduce a new technique that we believe may be of independent interest. The reason for this is that DDH works "natively" over the group $\mathbb{Z}_p$ where p is a super-polynomially large prime. Furthermore, we only have access to the $\mathbb{Z}_p$ elements in the exponent of a group generator g. Indeed, one can encode $0 \to g^0, 1 \to g^1$, and linear $\mathbb{Z}_2$ homomorphism will follow in the sense that after applying a linear function in the exponent, we obtain g^x, where $x \pmod 2$ is the desired $\mathbb{Z}_2$ output. This creates two obstacles: first, we need to be able to efficiently map $g^x \to x$, which means that x must come from a polynomially-bounded domain, and second that recovering x reveals more information than just $x \pmod 2$. We develop a new method to resolve this issue using *discrete Gaussian variables*. A technique that was used in the context of the LWE assumption but to the best of our knowledge not for DDH. We view this as an additional contribution to this thesis, which may find additional applications. In particular, we show that it can be used to enhance the key-dependent-message security properties of the well-known encryption scheme [BHHO08].

For more details on all of our contributions, see the technical overview in Section 4.2.

[4]Achieving optimal rate (or any rate above 1/2) seems to involve a "phase-transition" and should be viewed as more than a "constant factor" improvement. For example, OT beyond this threshold implies the existence of lossy trapdoor functions (see discussion in [DGI+19], Section 6.3). Therefore one could expect such a protocol to inherently be heavier on public-key operations.

[5]This is still a regime where LPN alone is not known to imply public-key encryption.

4.1.2 Related Work

The communication complexity of OT has been extensively studied throughout the decades. Here we present a brief overlook of previous works.

OT from Pseudorandom Correlations. A recent line of research studies the feasibility of efficiently extending OTs in a *silent* manner [BCG+19b, BCG+19a]. In these works, a setup phase is performed to distribute some *shares* between the parties. These shares can later be expanded into random OT correlations. In the most efficient scheme [BCG+19a] the setup phase can be performed in just two rounds assuming just a pseudorandom generator and an OT scheme. Using this scheme for performing the setup together with the standard transformations from random OT to chosen-input OT, [BCG+19a] shows that n independent instances of OT for s-bit strings can be performed with communication complexity $(2s+1)n + o(n)$. For bit OT, this yields a communication complexity $3n + o(n)$ bits.

Download rate-1 OT. We say that an OT protocol has a download rate-1 if the rate of the sender's message is asymptotically close to 1. OT protocols with download rate 1 were presented in [DGI+19, GHO20, CGH+21].However, these protocols do not achieve an upload rate-1, that is, the rate of the receiver's message is far from being 1. Moreover, it is not clear how we can extend these protocols to achieve upload rate 1.

Using rate-1 FHE. As mentioned before, optimal-rate OT can be achieved using the recent scheme for rate-1 fully homomorphic encryption (FHE) of [BDGM19, GH19] together with (semi-honest) circuit-privacy techniques for FHE (e.g. [BdMW16]). However, this can only be instantiated using LWE.

Laconic OT. Laconic OT [CDG+17, QWW18, GVW20, ABD+21a] is a flavour of two-round OT where the first message sent by the receiver is sublinear (ideally polylogarithmically) in the size of its input. However, by a simple information-theoretical argument, the sender's message has a size at least as large as the size of the sender's input. Note that, if this is not the case, then we would have an OT protocol with asymptotically better communication than an insecure OT protocol.

4.2 Techniques

4.2.1 Oblivious Transfer from Homomorphic Encryption

Our starting point is a textbook construction of oblivious transfer from simple homomorphic encryption schemes, such as ElGamal. For a cryptographic group $\mathbb{G} = \langle g \rangle$ of prime order p, recall that an ElGamal public key is of the form $\mathsf{pk} = (g, h = g^x) \in \mathbb{G}^2$, where $x \leftarrow_\$ \mathbb{Z}_p$ is the secret key. Ciphertexts are of the form $c = (c_1, c_2) = (g^r, h^r \cdot g^b)$, where $r \leftarrow_\$ \mathbb{Z}_p$ is uniformly random and $b \in \{0,1\}$ is the encrypted message. Given such a ciphertext c, the public key pk and two bits $m_0, m_1 \in \{0,1\}$, anyone can homomorphically compute a new ciphertext c' which is distributed identically to fresh encryption of m_b, by *homomorphically evaluating* the linear function $f(x) = (1-x) \cdot m_0 + x \cdot m_1 = (m_1 - m_0) \cdot x + m_0$ on the ciphertext c and rerandomizing the resulting ciphertext. Note that if $b \in \{0,1\}$ is a bit, then it holds that $f(b) = m_b$. This homomorphic evaluation can be achieved by computing

$$\begin{aligned} c_1' &\leftarrow g^{r^*} \cdot c_1^{m_1 - m_0} \\ c_2' &\leftarrow h^{r^*} \cdot c_2^{m_1 - m_0} \cdot g^{m_0}, \end{aligned}$$

where $r^* \leftarrow_\$ \mathbb{Z}_p$ is chosen uniformly random. Note that it holds that

$$\begin{aligned} c_1' &= g^{r^* + r \cdot (m_1 - m_0)} \\ c_2' &= h^{r^* + r \cdot (m_1 - m_0)} \cdot g^{(m_1 - m_0) \cdot b + m_0} = h^{r^* + r \cdot (m_1 - m_0)} \cdot g^{m_b}. \end{aligned}$$

Since $r^* \leftarrow_\$ \mathbb{Z}_p$ is chosen uniformly random, it holds that $r' = r^* + r \cdot (m_1 - m_0)$ is distributed uniformly random and we can conclude that $c' = (c_1', c_2')$ is distributed identical to a fresh encryption of m_b. Since c' does not reveal more than the function value $f(b) = m_b$, we call the above homomorphic evaluation procedure to function private.

This immediately implies an OT protocol: An OT-receiver holding a choice-bit $b \in \{0,1\}$ generates a pair $(\mathsf{pk}, \mathsf{sk})$ of ElGamal public and secret keys, encrypts the bit b under pk and sends the resulting ciphertext to the OT-sender. The OT-sender, holding messages m_0, m_1, homomorphically computes a ciphertext c' encrypting m_b and sends c' back to the OT-receiver, who decrypts c' to m_b. Security against semi-honest senders follows from the IND-CPA security of ElGamal, whereas security against semi-honest receivers follows from the function privacy property established above.

4.2.2 Download-Rate Optimal String OT

While the above OT protocol is simple and efficient, it suffers from a very poor communication rate. While the receiver's message encrypts just a single bit, he needs to send 4 group elements, whereas the sender sends 2 group elements, each of size $\mathsf{poly}(\lambda)$.

Döttling et al. [DGI+19] proposed a compression technique for *batched ElGamal ciphertexts* based on the share-conversion technique of [BGI16]. A batched ElGamal ciphertext is of the form $\mathbf{c} = (c_0, c_1, \ldots, c_\ell) = (g^r, h_1^r \cdot g^{b_1}, \ldots, h_\ell^r \cdot g^{b_\ell})$, where $\mathsf{pk} = (g, h_1, \ldots, h_\ell)$ is the corresponding public key and $\mathsf{sk} = (s_1, \ldots, s_\ell)$ with $h_i = g^{s_i}$ is the secret key. The compression technique of [DGI+19] keeps c_0 and compresses each of the $c_1, \ldots, c_\ell$ into just a single bit. The idea is instead of sending each $c_i \in \mathbb{G}$ (for $i \geq 1$) in full, to first compute the distance d to the next pseudorandom *break-point* in $\mathbb{G}$, and then only send its parity $d \mod 2$. The break points $\mathcal{P} \subseteq \mathbb{G}$ are the set of all points $h \in \mathbb{G}$ satisfying $\mathsf{PRF}_K(h) = 0^t$, where $\mathsf{PRF} : \mathbb{G} \to \{0,1\}^t$ is a pseudorandom function with a range of size $2^t = \mathsf{poly}(\lambda)$. Thus, the distance $d = d(c_i)$ of a group element c_i to the nearest breakpoint is the smallest non-negative d such that $c_i \cdot g^d \in \mathcal{P}$. Given that neither c_i nor $c_i \cdot g^{-1}$ is a breakpoint, we can recover the bit b_i from $c_0 = g^r, \beta_= d(c_i) \mod 2$ and the secret key component s_i. It was shown in [BBD$^+$20] that for a given ciphertext $c = (c_0, c_1, \ldots, c_\ell)$, the PRF-key K can be (efficiently) chosen such that all c_i are good, in the sense that neither c_i nor $c_i \cdot g^{-1}$ is a breakpoint. This ensures that a receiver can recover the $b_1, \ldots, b_\ell$ from $c' = (K, c_0, \beta_1, \ldots, \beta_\ell)$, where $\beta_i = d(c_i) \mod 2$. Since all the β_i are bits, such a compressed ciphertext only has additive size-overhead consisting of K, c_0. For a sufficiently large ℓ, this fixed overhead becomes insignificant and the ciphertext rate approaches 1.

The compressed batched ElGamal we've outlined leads to a batch bit-oblivious transfer protocol with *download-rate 1*: The receiver generates a key-pair pk, sk for batched ElGamal, and encrypts his choice-bits $b_1, \ldots, b_\ell$ into

$$\mathbf{c}_1 = \mathsf{Enc}_{\mathsf{pk}}(b_1, 0, \ldots, 0), \ldots, \mathbf{c}_\ell = \mathsf{Enc}_{\mathsf{pk}}(0, \ldots, 0, b_\ell),$$

i.e. $\mathbf{c}^{(i)}$ encrypts a vector which is b_i in index i and 0 everywhere else. The OT-receiver now sends $\mathsf{pk}, \mathbf{c}_1, \ldots, \mathbf{c}_\ell$ to the OT-sender, whose input are messages $(m_{1,0}, m_{1,1}), \ldots, (m_{\ell,0}, m_{\ell,1})$. Using circuit private homomorphic evaluation, the sender computes ciphertexts $\mathbf{c}'_1, \ldots, \mathbf{c}'_\ell$ encrypting $(m_{1,b_1}, 0, \ldots, 0)$, $\ldots, (0, \ldots, 0, m_{\ell,b_\ell})$. Homomorphically computing the sum of the ciphertexts $\mathbf{c}'_1, \ldots, \mathbf{c}'_\ell$, we obtain a ciphertext $\mathbf{c}'$ encrypting $(m_{1,b_1}, \ldots, m_{\ell,b_\ell})$. Finally, compressing $\mathbf{c}'$ with the compression technique outlined above we obtain a compressed ciphertext $\bar{\mathbf{c}} = (K, c_0, \beta_1, \ldots, \beta_\ell)$ which the OT-sender sends back to the OT-receiver, who can decrypt $(m_{1,b_1}, \ldots, m_{\ell,b_\ell})$.

Note that the size of the sender's message $\bar{\mathbf{c}}$ in this batch OT-protocol is $\mathsf{poly}(\lambda) + \ell$, which means that the *amortized communication cost* per bit-OT approaches 1 bit, and is therefore asymptotically optimal. Even in terms of concrete complexity, this seems hard to beat, as the only additional information sent by the sender are the PRF key K and the ciphertext header c_0.

However, in terms of the upload rate, i.e. in terms of the size of the receiver's message, this protocol performs poorly. Specifically, to encrypt ℓ bits $b_1, \ldots, b_\ell$, the receiver needs to send ciphertexts $\mathbf{c}_1, \ldots, \mathbf{c}_\ell$ of total size $\ell^2 \cdot \mathsf{poly}(\lambda)$, which has a worse dependence on ℓ than just repeating the simple protocol from the last paragraph ℓ times.

Clearly, we need a mechanism to compress the receiver's message. Applying the same ElGamal compression technique for the sender's message quickly runs into problems: Once an ElGamal ciphertext is compressed, the scheme loses its homomorphic capabilities, i.e. we cannot perform any further homomorphic operations on compressed ciphertexts and currently we don't know if it is possible to publicly *decompress* such ciphertexts into "regular" ElGamal ciphertexts.

4.2.3 Our Approach: Recrypting the Receiver's Message

Instead, our approach will be to encrypt the receiver's message under a different encryption scheme, specifically one which achieves a ciphertext rate approaching 1 but at the same time can be decrypted by the homomorphic capabilities of batched ElGamal. Specifically, the decryption procedure of this encryption scheme should be a linear function in the secret key. We can get an encryption scheme which almost fulfills these requirements from the Learning Parity with Noise (LPN) assumption. The LPN assumption states that for a random $m \times n$ matrix $\mathbf{A} \leftarrow_\$ \mathbb{Z}_2^{m \times n}$, a random vector $\mathbf{s} \leftarrow_\$ \mathbb{Z}_2^n$ and a ρ-Bernoulli distributed [6] $\mathbf{e} \in \mathbb{Z}_2^m$, it holds that

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

where $\mathbf{u} \leftarrow_\$ \mathbb{Z}_2^m$ is chosen uniformly at random. This gives rise to the following simple symmetric-key encryption scheme with *approximate correctness*: Assume that $\mathbf{A}$ is a fixed public parameter, the secret key is a uniformly random $\mathbf{s} \leftarrow_\$ \mathbb{Z}_2^n$. To encrypt a message $m \in \mathbb{Z}_2^m$, we compute a ciphertext $\mathbf{d} \leftarrow \mathbf{A}\mathbf{s}+\mathbf{e}+\mathbf{m}$, where $\mathbf{e} \in \mathbb{Z}_2^m$ is chosen via a ρ-Bernoulli distribution. To decrypt such a ciphertext, we compute $\mathbf{m}' \leftarrow \mathbf{d} - \mathbf{A} \cdot \mathbf{s}$.

Note that this scheme is only approximately correct in the sense that it holds that $\mathbf{m}' = \mathbf{m} + \mathbf{e}$, i.e. in most coordinates $\mathbf{m}'$ is identical to $\mathbf{m}$, but only in few coordinates $\mathbf{m}'$ and $\mathbf{m}$ differ. Furthermore, the one-time security of this encryption scheme follows from the LPN assumption.

The high-level strategy to use this symmetric key encryption scheme is now as follows: Assume the matrix $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ is known to both the sender and the receiver. In the actual protocol this matrix will be chosen by the receiver, and the communication cost of sending $\mathbf{A}$ will be amortized by reusing $\mathbf{A}$ many times.

The OT-receiver chooses a symmetric key $\mathbf{s} \leftarrow_\$ \mathbb{Z}_2^n$ uniformly at random and encrypts his vector of choice bits $\mathbf{b} = (b_1, \dots, b_\ell)$ to $\mathbf{d} = \mathbf{A}\mathbf{s} + \mathbf{e} + \mathbf{b}$ (where again, $\mathbf{e} \in \mathbb{Z}_2^\ell$ is ρ-Bernoulli distributed). Furthermore, the receiver will encrypt the LPN secret under ElGamal, i.e. he encrypts $\mathbf{s}$ to $\mathbf{c} = \mathsf{Enc}(\mathsf{pk}, \mathbf{s})$. For the moment, assume that $\mathbf{s}$ is encrypted bit-wise with standard ElGamal rather than batched ElGamal. The OT-receiver now sends the ElGamal public key pk and the ciphertexts $\mathbf{c}$ and $\mathbf{d}$ to the OT-sender.

Now, given these values, the sender can homomorphically decrypt the $\mathbf{d}$ into ElGamal, effectively *key-switching* from the ciphertext $\mathbf{d}$ into an ElGamal ciphertext. Concretely: The sender homomorphically evaluates the linear function $f(\mathbf{x}) = \mathbf{d} - \mathbf{A}\mathbf{x}$ on the ElGamal ciphertext $\mathbf{c} = \mathsf{Enc}(\mathsf{pk}, \mathbf{s})$. This produces an ElGamal encryption $\mathbf{c}'$ encrypting $f(\mathbf{s}) = \mathbf{d} - \mathbf{A}\mathbf{s} = \mathbf{b} + \mathbf{e} = \mathbf{b}'$. In other words, the OT-sender has now obtained an ElGamal encryption of a vector $\mathbf{b}'$ which agrees with $\mathbf{b}$ in most locations.

The high-level idea is now to let the OT-sender use this ciphertext $\mathbf{c}'$ as the encryption of the receiver's choice bits and proceed as in the ElGamal-based OT protocol above. If we were to naively use $\mathbf{c}'$ in this way, the receiver would obtain the correct output m_{i,b_i} in locations where $\mathbf{b}$ and $\mathbf{b}'$ agree but would get the wrong output $m_{i,1-b_i}$ in locations where $\mathbf{b}$ and $\mathbf{b}'$ disagree. While there certainly are applications in which a small number of faulty locations are tolerable, in general, this leads to insecure protocols.

There is, however, another issue with this approach. In this paragraph, we have implicitly assumed that ElGamal is homomorphic for linear functions modulo 2. However, since the group we implement ElGamal over is of large prime order p, when we evaluate linear functions such as $f(\mathbf{x}) = \mathbf{d} - \mathbf{A}\mathbf{x}$ over a ciphertext encrypting a $\mathbf{s} \in \{0,1\}^n$, the result of this evaluation is *not* reduced modulo 2, and the resulting ciphertext in fact encrypts $f(\mathbf{s})$ as an integer. This does not cause major problems in

[6] i.e. every component of e_i of $\mathbf{e}$ is independently 0 with probability $1 - \rho$ and 1 with probability ρ

terms of correctness, as this integer will still be small (at most of the size m), and hence decryption will still be efficient.

However, this does cause major problems in terms of sender privacy, as we can only guarantee sender privacy for receiver messages that are guaranteed to encrypt a bit $b \in \{0,1\}$.

For now, we will bypass this problem by relying on a homomorphic encryption scheme which is in fact homomorphic over $\mathbb{Z}_2$ (rather than $\mathbb{Z}_p$), offers function privacy for linear functions modulo 2 and is compatible with ciphertext compression. Such encryption can in fact be constructed from the Quadratic Residuosity assumption [DGI+19].

Another small issue we haven't addressed here is that the compression mechanisms for the sender and the receiver are somewhat orthogonal, in the sense that the sender's message is compressed by compressing a batched ElGamal ciphertext (which generally does not allow homomorphic evaluation across different components), whereas the receiver's compression strategy requires the homomorphic evaluation of linear functions with multiple (i.e. vector-valued) inputs. In the main body (Section 4.7) we will show a tradeoff which allows us to reconcile these requirements, leading to a batch OT protocol with an overall rate-1.

We will first discuss how to deal with the issue of errors in the key-switched ciphertext and then return to the issue of implementing our approach with ElGamal instead of QR-based encryption.

4.2.4 Dealing with LPN Errors

To deal with the LPN errors in the key-switched ciphertext $\mathbf{c}'$, we will pursue the following high-level strategy: The sender will introduce additional masking on the receiver's output, which can only be removed in error-free locations. This masking effectively erases the receiver's output in locations in which the receiver's output is corrupted.

To communicate the correct outputs in the locations with errors, the parties will rely on an additional protocol which is run in parallel. Given that the number of errors is sufficiently small, the communication cost of this additional protocol will be insubstantial and not affect the overall asymptotic rate.

We will first address the problem of erasing the receiver's output in corrupted locations. First, observe that the receiver knows the locations with errors (i.e. the support of the error vector $\mathbf{e}$). Assume that the LPN error vector $\mathbf{e}$ has a fixed hamming weight $t \approx \rho m$, and note that hardness of fixed-weight LPN follows routinely from the hardness of Bernoulli LPN[7]. A t-puncturable pseudorandom function [BGI14, BCG+19b] is a pseudorandom function [GGM84] which supports punctured keys. That is, given a PRF key K and t inputs $x_1, \ldots, x_t$, we can efficiently compute a *punctured key* K' of size $t \cdot \mathsf{poly}(\lambda)$ which allows evaluating the PRF on all inputs *except* $x_1, \ldots, x_t$. Furthermore, the key K' does not reveal the function values at $x_1, \ldots, x_t$, i.e. $\mathsf{PRF}(K, x_1), \ldots, \mathsf{PRF}(K, x_t)$ are pseudorandom given the punctured key K'.

The approach to erase the receiver's outputs in erroneous locations is now as follows. The sender chooses a PRF key K and masks both $m_{i,0}$ and $m_{i,1}$ with $\mathsf{PRF}(K, i)$, i.e. instead of using $(m_{i,0}, m_{i,1})$

[7]See e.g. [Döt15, BCG+19b]

as OT-inputs, he uses $m'_{i,0} = m_{i,0} \oplus \mathsf{PRF}(K,i)$ and $m'_{i,1} = m_{i,1} \oplus \mathsf{PRF}(K,i)$. Assuming that the sender can somehow communicate a punctured key K' which is punctured at the locations $i_1, \dots, i_t$ of the errors (i.e. $\mathbf{e}_{i_j} = 1$ and $\mathbf{e}$ is 0 everywhere else), the receiver will be able to remove the mask from error-free locations by computing $m_{i,b_i} = m'_{i,b_i} \oplus \mathsf{PRF}(K',i)$. In the erroneous locations, however, $m_{i,1-b_i}$ will be hidden from the view of the receiver as $\mathsf{PRF}(K,i)$ is pseudorandom even given the punctured key K'.

How can we communicate the punctured key K' to the receiver with a small communication cost in such a way that the sender does not learn the error locations $i_1, \dots, i_t$? This could be achieved generically by relying on the punctured PRF construction of [BGI14] and transferring keys using a sublinear private information retrieval (PIR) scheme [CGKS95, DGI+19]. However, recently [BCG+19b] provided a protocol to achieve this task very efficiently via a two-round protocol communicating only $t\mathsf{poly}(\lambda)$ bits. In the main body (Section 4.6), we will refer to this primitive as *co-PIR*, since effectively it allows to communicate of a large pseudorandom database to a receiver except in a few locations chosen by the receiver.

Finally, to communicate the correct outputs to the receiver in the locations with errors, we will in fact rely on a two-message PIR scheme with polylogarithmic communication. Such schemes are known e.g. from LWE [BV11] and were recently constructed from a wide variety of assumptions [DGI+19], such as DDH and QR. The idea is as follows: For each error location i_j the receiver sends an additional OT message $OT_1(b_{i_j})$ using an off-the-shelf low-rate OT protocol (e.g. the basic ElGamal-based protocol sketched above), as well as a PIR message $PIR_1(i_j)$. The sender speculatively completes this OT protocol for each index i (since the index i_j is not known to the sender), collects his OT responses in a database of size ℓ, runs the PIR sender algorithm on this database, and sends the response back to the receiver. The receiver will now be able to recover the correct OT_2 message via PIR, complete the OT and recover $m_{i_j,b_{i_j}}$. We remark that for this protocol to be secure against semi-honest senders, we need a PIR protocol with sender privacy. However, e.g. the protocols provided in [DGI+19] readily have this feature.

Carefully putting all these components together, we obtain a batch bit-OT protocol with rate-1, for both the sender and the receiver.

4.2.5 Emulating Small Subgroups

We now return to the issue that ElGamal does not provide function privacy for linear functions modulo 2. Recall that the issue essentially boils down to the fact that the plaintext space of ElGamal is *natively* $\mathbb{Z}_p$, and when we encode messages in the least significant bits, i.e. encoding a bit b as g^b, then for all practical purposes homomorphic evaluations of linear functions with $\{0,1\}$ coefficients are over $\mathbb{Z}_2$, i.e. the resulting ciphertext encodes the result of the function evaluation *without* reduction modulo 2.

From an algebraic perspective, this problem is rooted in the fact that since p is prime, $\mathbb{Z}_p$ has no non-trivial subgroup, i.e. it just does not support modular reductions with respect to anything else than p.

To approach this problem, we will take inspiration from the domain of lattice cryptography [Reg05]. There, messages are typically encoded in the high order bits of group elements, i.e. to encode b in $\mathbb{Z}_p$, we would like to encode it as $b \cdot \frac{p}{2}$. However, since p is odd, first have to round $\frac{p}{2}$ to the nearest integer in order to get a proper $\mathbb{Z}_p$ element, i.e. we encode b via $b \cdot \left\lceil \frac{p}{2} \right\rceil$. If we could encode b with respect to $\frac{p}{2} \notin \mathbb{Z}_p$, we would get a subgroup of order 2, i.e. for bits $b, b' \in \{0,1\}$ it holds that

$$\left(b \cdot \frac{p}{2} + b' \cdot \frac{p}{2}\right) \bmod p = (b + b' \bmod 2) \cdot \frac{p}{2}.$$

However, once we round $\frac{p}{2}$ to the next integer, we get essentially the same problem as before: If we perform group operations on $b \left\lceil \frac{p}{2} \right\rceil$ and $b' \left\lceil \frac{p}{2} \right\rceil$, then the rounding errors start to accumulate information about b and b' which is cannot be obtained from $b + b' \bmod 2$. Specifically

$$\begin{aligned} b \left\lceil \frac{p}{2} \right\rceil + b' \left\lceil \frac{p}{2} \right\rceil \bmod p &= b\left(\frac{p}{2} + \frac{1}{2}\right) + b'\left(\frac{p}{2} + \frac{1}{2}\right) \bmod p \\ &= (b + b' \bmod 2)\frac{p}{2} + (b + b')\frac{1}{2} \bmod p. \end{aligned}$$

Thus, now the least significant bit of $b \left\lceil \frac{p}{2} \right\rceil + b' \left\lceil \frac{p}{2} \right\rceil \bmod p$ e.g. leaks if $b = b' = 1$, something which cannot be learned from $b + b' \bmod 2$.

Consequently, at first glance the idea of encoding a bit b in the "high-order" bits of a $\mathbb{Z}_p$ element seems ineffective. However, the lattice toolkit still has more to offer. In particular, in the context of sampling discrete Gaussians from lattices, Peikert [Pei10] considered a technique called *randomized rounding*. The basic idea is, to give a real number $r \in \mathbb{R}$ to not always round to the same value e.g. $\lceil r \rceil$, but to sample an integer z close to r. In [Pei10], this distribution is a discrete gaussian Z on $\mathbb{Z}$ centered at r, i.e. the expectation of Z is r. Such a discrete gaussian is parametrized by a gaussian parameter σ, which essentially controls the standard deviation of the discrete gaussian. We denote Z by $\lceil r \rfloor_\sigma$.

Now, given any two $r, r' \in \mathbb{R}$ and $\sigma_1, \sigma_2 > \omega(\sqrt{\log(\lambda)})$ (more generally the *smoothing parameter* of $\mathbb{Z}$), Peikert [Pei10] shows that

$$\lceil r \rfloor_{\sigma_1} + \lceil r' \rfloor_{\sigma_2} \approx_s \lceil r + r' \rfloor_{\sqrt{\sigma_1^2 + \sigma_2^2}}.$$

In other words, while $\lceil r \rfloor_{\sigma_1} + \lceil r' \rfloor_{\sigma_2}$ and $\lceil r + r' \rfloor_{\sqrt{\sigma_1^2+\sigma_2^2}}$ are note the same, they are statistically close. This means that anything that can be learned from $\lceil r \rfloor_{\sigma_1} + \lceil r' \rfloor_{\sigma_2}$ could have as well been learned from $\lceil r + r' \rfloor_{\sqrt{\sigma_1^2+\sigma_2^2}}$! While this comes at the expense of an increase "error" term with parameter $\sqrt{\sigma_1^2 + \sigma_2^2}$, this *additive error* is very small (of size approx σ) controlling the growth of this error term can be handled by standard techniques.

Returning to our goal of emulating small subgroups in $\mathbb{Z}_p$, our approach follows almost instantly: Instead of encoding a bit $b \in \mathbb{Z}_2$ as $b \cdot \left\lceil \frac{p}{2} \right\rceil$, we will encode it as $\left\lceil b \cdot \frac{p}{2} \right\rfloor_\sigma$ (for a $\sigma > \omega(\sqrt{\log(\lambda)})$).

For $b, b' \in \{0,1\}$ this ensures that

$$\left\lceil b \cdot \frac{p}{2} \right\rfloor_\sigma + \left\lceil b' \cdot \frac{p}{2} \right\rfloor_\sigma \bmod p \approx_s \left\lceil (b + b' \bmod 2) \cdot \frac{p}{2} \right\rfloor_{\sqrt{2}\sigma} \bmod p.$$

Thus, we have ensured that $\lceil b \cdot \frac{p}{2} \rfloor_\sigma + \lceil b' \cdot \frac{p}{2} \rfloor_\sigma \bmod p$ does not leak more information than $b + b' \bmod 2$.

Function-Private Evaluation for ElGamal We will now briefly discuss how this idea leads to a modulo 2 function private homomorphic evaluation procedure for ElGamal. Say we have two ElGamal ciphertexts $c_1 = (g^{r_1}, h^{r_1} \cdot g^{b_1})$ and $c_2 = (g^{r_2}, h^{r_2} \cdot g^{b_2})$ for a public key $\mathsf{pk} = (g, h)$ and we want to homomorphically evaluate the function $f(x_1, x_2) = a_1 x_1 + a_2 x_2 \mod 2$ (for $a_1, a_2 \in \{0,1\}$) on this pair of ciphertexts. In the first step, we *randomly encode* the function f as

$$f'(x_1, x_2) = x_1 \cdot \left\lceil a_1 \frac{p}{2} \right\rfloor_\sigma + (1 - x_1) \cdot \lceil 0 \rfloor_\sigma + x_2 \cdot \left\lceil a_2 \frac{p}{2} \right\rfloor_\sigma + (1 - x_2) \cdot \lceil 0 \rfloor_\sigma,$$

noting that this is still a linear function (chosen from a distribution). Homomorphically evaluating f' on the ciphertexts $c_, c_2$ we obtain a ciphertext c' encrypting

$$\begin{aligned} f'(b_1, b_1) &= b_1 \cdot \left\lceil a_1 \frac{p}{2} \right\rfloor_\sigma + (1 - b_1) \cdot \lceil 0 \rfloor_\sigma + b_1 \cdot \left\lceil a_2 \frac{p}{2} \right\rfloor_\sigma + (1 - b_1) \cdot \lceil 0 \rfloor_\sigma \\ &= \left\lceil b_1 a_1 \frac{p}{2} \right\rfloor_\sigma + \left\lceil b_1 a_2 \frac{p}{2} \right\rfloor_\sigma \\ &\approx_s \left\lceil (b_1 a_1 + b_1 a_2 \bmod 2) \frac{p}{2} \right\rfloor_{\sqrt{2}\sigma}. \end{aligned}$$

In other words, this ciphertext could have been simulated knowing only the function result $f(b_1, b_1) = b_1 a_1 + b_1 a_2 \bmod 2$, establishing that this homomorphic evaluation procedure is a function private.

One aspect to note is that while the messages b_1, b_1 are encoded in c_1, c_2 in the "low-order-bits" via g^{b_1} and g^{b_2}, the function result $f(b_1, b_2)$ encrypted in c' is encoded in the high order bits, i.e. it is encoded as $\approx g^{f(b_1, b_2) \frac{p}{2}}$. This makes it necessary to change the decryption procedure: Let $c' = (c'_1, c'_2)$ and s be the secret key. To decrypt c' we compute $f = c'_2 \cdot (c'_1)^{-s} \approx_s g^{\lceil f(s_1, s_2) \cdot \frac{p}{2} \rfloor}$, we test if f is close to $g^0 = 1$ or $g^{\lceil p/2 \rceil}$. This recovers $f(s_1, s_2)$, as the error introduced by the rounding operation is of size at most $\mathsf{poly}(\lambda)$ via standard gaussian tail bounds.

Finally, we remark this "high-order-bit" encoding is still compatible with ElGamal ciphertext compression, i.e. we can still compress homomorphically evaluated batch ElGamal ciphertexts down asymptotically optimal size, using a slightly different compression mechanism. This mechanism is discussed in Section 4.5. We expect this technique to have additional applications. As one immediate application, it allows to upgrade of the key-dependent message secure encryption scheme of Boneh et al. [BHHO08] to support arbitrary linear functions modulo 2.

4.3 Definitions

In this chapter, we consider the two-message oblivious transfer (OT) with an overall (almost) optimal rate; where the sender has input $(m_0, m_1) \in \{0,1\}^2$ and the receiver a choice bit $b \in \{0,1\}$. In the end, the receiver learns the bit m_b and nothing else; the sender learns nothing about b. We define the OT in the plain model as follows.

Definition 4.3.1 (Two-message OT). A two-message OT protocol between a sender and a receiver can be defined as a tuple of three PPT algorithms $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$. Let λ be the security parameter and $k = \mathsf{poly}(\lambda)$. The receiver computes $(\mathsf{ot}_1, \mathsf{st}) \leftarrow \mathsf{OTR}(1^\lambda, \mathbf{b})$ with his input $\mathbf{b} = (b_1, \dots, b_k) \in \{0,1\}^k$ and sends ot_1 to the sender. The sender computes $\mathsf{ot}_2 \leftarrow \mathsf{OTS}(1^\lambda, \mathsf{ot}_1, (\mathbf{m}_0, \mathbf{m}_1))$ where $(\mathbf{m}_0, \mathbf{m}_1) = ((m_{0,1}, \dots, m_{0,k})(m_{1,1}, \dots, m_{1,k}) \in (\{0,1\}^k)^2$ and sends to the receiver ot_2. At the end, the receiver decodes the message to get $\mathbf{m_b} = (m_{b_1,1}, \dots, m_{b_k,k}) \leftarrow \mathsf{OTD}(\mathsf{ot}_2, \mathsf{st})$.

In terms of security, OT should implement the following functionality.

OT functionality. The functionality $\mathcal{F}_{\mathsf{OT}}$ is parametrized by a integer $k = \mathsf{poly}(\lambda)$ and works as follows:

- **Receiver phase.** R sends $\mathbf{b}$ to $\mathcal{F}_{\mathsf{OT}}$ where $\mathbf{b} \in \{0,1\}^k$.
- **Sender phase.** S sends $(\mathbf{m}_0, \mathbf{m}_1)$ to $\mathcal{F}_{\mathsf{OT}}$ where $\mathbf{m}_0, \mathbf{m}_1 \in \{0,1\}^k$. $\mathcal{F}_{\mathsf{OT}}$ sends $\{\mathbf{m}_{b_i,i}\}_{i\in[k]}$ to R.

4.3.1 Distributed GGM-PPRF Correlation

Let $\mathsf{PPRF}_{\mathsf{GGM}} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{EvalPunct})$ be the GGM-PPRF scheme based on [GGM86]. The distributed GGM-PPRF correlation functionality [BCG+19a] considers two parties: A receiver with input $\alpha \in \{0,1\}^\ell$ and a sender with input $\beta \in \mathbb{F}_{p^r}$ and a GGM-PPRF key K. The functionality outputs a punctured key K_α and a hardwired value $\beta - \mathsf{PPRF.Eval}(\mathsf{K}, \alpha)$ to the receiver. We now present the formal definition of the functionality.

Distributed GGM-PPRF correlation functionality. The functionality $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ is parametrized by integers $\ell, p, r \in \mathbb{N}$. Moreover, let $\mathsf{PPRF}_{\mathsf{GGM}} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{EvalPunct})$ be the GGM PPRF scheme with input space $\{0,1\}^\ell$ and output space $\mathbb{F}_{p^r}$. The functionality works as follows:

- **Receiver phase.** R sends α to $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ where $\alpha \in \{0,1\}^\ell$.
- **Sender phase.** S sends (β, K) to $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ where $\beta \in \mathbb{F}_{p^r}$ and $\mathsf{K} \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$. $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ sends $\mathsf{K}_\alpha \leftarrow \mathsf{PPRF.Puncture}(\mathsf{K}, \alpha)$ and $\gamma \leftarrow \beta - \mathsf{PPRF.Eval}(\mathsf{K}, \alpha)$ to R.

A protocol that implements the functionality $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ is presented in [BCG+19a]. The protocol uses a pseudorandom generator (PRG) and an oblivious transfer (OT)[8] protocol in a black-box way. Moreover, security is proven against semi-honest adversaries. Finally, the protocol presented runs in two rounds (assuming that the OT runs in two rounds) and achieves communication complexity of $\mathsf{poly}(\lambda, \ell)$.

For convenience, we will denote such a protocol by $\mathsf{PPRF\text{-}GGM} = (\mathsf{PPRF\text{-}GGM.R_1}, \mathsf{PPRF\text{-}GGM.S}, \mathsf{PPRF\text{-}GGM.R_2})$ where:

- $\mathsf{PPRF\text{-}GGM.R_1}(\alpha)$ receives as input $\alpha \in \{0,1\}^{\ell}$. It outputs a message $\mathsf{pprf\text{-}ggm_1}$ and a state st.
- $\mathsf{PPRF\text{-}GGM.S}(\beta, \mathsf{K}, \mathsf{pprf\text{-}ggm_1})$ receives as input $\beta \in \mathbb{F}_{p^r}$, a key $\mathsf{K} \leftarrow \mathsf{PPRF.KeyGen}(1^{\lambda})$ and a message $\mathsf{pprf\text{-}ggm_1}$. It outputs $\mathsf{pprf\text{-}ggm_2}$.
- $\mathsf{PPRF\text{-}GGM.R_2}(\mathsf{st}, \mathsf{pprf\text{-}ggm_2})$ receives as input a state st and a message $\mathsf{pprf\text{-}ggm_2}$. It outputs a punctured key K_{α} and a value $\gamma \in \mathbb{F}_{p^r}$.

Using the two-round OT scheme of [PVW08], we can obtain a black-box construction for distributed GGM-PPRF correlation scheme under the LWE, DDH or QR assumptions.

4.4 Compression-friendly Subgroup Emulation via Gaussian Rounding

We will now provide our new subgroup emulation technique. We first define the gaussian rounding functionality.

Definition 4.4.1. Let $\sigma > 0$. For any $x \in \mathbb{R}$, the gaussian rounding $\lceil x \rfloor_{\sigma}$ is a random variable supported on $\mathbb{Z}$ defined by

$$\lceil x \rfloor_{\sigma} = x + D_{\mathbb{Z}-x,\sigma}.$$

In other words, $\lceil x \rfloor_{\sigma}$ is a discrete gaussian centred on $x \in \mathbb{R}$ but supported on $\mathbb{Z}$.

We will use the following convolution lemma which provides a *simulation property* for gaussian rounding.

Lemma 4.4.1. *Let $\varepsilon > 0$ be bounded by a sufficiently small constant and let $\sigma_1, \sigma_2 \geq \eta_{\varepsilon}(\mathbb{Z})$. Then it holds for all $x, y \in \mathbb{R}$ that*

$$\lceil x \rfloor_{\sigma_1} + \lceil y \rfloor_{\sigma_2} \approx_s \lceil x + y \rfloor_{\sqrt{\sigma_1^2 + \sigma_2^2}}.$$

It immediately follows from Lemma 4.4.1 that it holds for every integer $p \geq 2$ that

$$\lceil x \rfloor_{\sigma_1} + \lceil y \rfloor_{\sigma_2} \mod p \approx_s \lceil x + y \rfloor_{\sqrt{\sigma_1^2 + \sigma_2^2}} \mod p.$$

[8]The OT protocol is not required to have overall rate 1.

Proof. The lemma follows routinely from Corollary 1.3.3, by definition of $\lceil \cdot \rfloor_\sigma$ it holds that

$$\begin{aligned}\lceil x \rfloor_{\sigma_1} + \lceil y \rfloor_{\sigma_2} &= x + y + D_{\mathbb{Z}-x,\sigma_1} + D_{\mathbb{Z}-y,\sigma_2} \\ &\approx_s x + y + D_{\mathbb{Z}-x-y,\sigma_3} \\ &= \lceil x + y \rfloor_{\sigma_3}.\end{aligned}$$

□

Lemma 4.4.2. *Let $p > q \geq 2$ be integers with $q \leq 2^k$, and let $\sigma > \eta_\varepsilon(\mathbb{Z})$ for a negligible ε. Let $f : \mathbb{Z}_q^n \to \mathbb{Z}_q$ be given by $f(x_1, \ldots, x_n) = \sum_{i=1}^n a_i x_i + c$ for $a_1, \ldots, a_n, c \in \mathbb{Z}_q$. Define the randomized function $\hat{f} : \{0,1\}^{nk} \to \mathbb{Z}_p^n$ via*

$$\hat{f}(x_{1,1}, \ldots, x_{n,k}) = \sum_{i=1}^n \sum_{j=1}^k \left(x_{i,j} \cdot \left\lceil 2^j \cdot \frac{p}{q} a_i \right\rfloor_\sigma + (1 - x_{i,j}) \lceil 0 \rfloor_\sigma \right) + \left\lceil \frac{p}{q} c \right\rfloor_\sigma .$$

Then it holds for all $x_{1,1}, \ldots, x_{n,k} \in \{0,1\}$ that

$$\hat{f}(x_{1,1}, \ldots, x_{n,k}) \approx_s \left\lceil \frac{p}{q} \cdot f\left(\sum_{j=1}^k x_{1,j} 2^j, \ldots, \sum_{j=1}^k x_{n,j} 2^j \right) \right\rfloor_{\sqrt{2nk+1}\sigma} .$$

Proof. It holds routinely that

$$\begin{aligned}\hat{f}(x_{1,1}, \ldots, x_{n,k}) &= \sum_{i=1}^n \sum_{j=1}^k \left(x_{i,j} \cdot \left\lceil 2^j \cdot \frac{p}{q} a_i \right\rfloor_\sigma + (1 - x_{i,j}) \lceil 0 \rfloor_\sigma \right) + \left\lceil \frac{p}{q} c \right\rfloor_\sigma \\ &\approx_s \sum_{i=1}^n \sum_{j=1}^k \left\lceil x_{i,j} 2^j \cdot \frac{p}{q} a_i \right\rfloor_{\sqrt{2}\sigma} + \left\lceil \frac{p}{q} c \right\rfloor_\sigma \qquad (4.1) \\ &\approx_s \left\lceil \sum_{i=1}^n \sum_{j=1}^k x_{i,j} 2^j \cdot \frac{p}{q} a_i + \frac{p}{q} c \right\rfloor_{\sqrt{2nk+1}\sigma} \qquad (4.2) \\ &= \left\lceil \frac{p}{q} \cdot \left(\sum_{i=1}^n a_i \left(\sum_{j=1}^k x_{i,j} 2^j \right) + c \right) \right\rfloor_{\sqrt{2nk+1}\sigma} \\ &= \left\lceil \frac{p}{q} \cdot f\left(\sum_{j=1}^k x_{1,j} 2^j, \ldots, \sum_{j=1}^k x_{n,j} 2^j \right) \right\rfloor_{\sqrt{2nk+1}\sigma} ,\end{aligned}$$

where in equations (4.1) and (4.2) we have used Lemma 4.4.1. □

4.5 Rate-1 Circuit-Private Linearly Homomorphic Encryption

In this section, we define circuit-private LHE and present constructions based on LWE, DDH or QR. All constructions achieve rate 1.

Definition 4.5.1. A (packed) *linearly homomorphic encryption* (LHE) scheme LHE over a finite group $\mathbb{G}$ is composed by a tuple of algorithms $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Shrink}, \mathsf{DecShrink})$ such that:

- $\mathsf{KeyGen}(1^\lambda, k)$ takes as input a security parameter λ and $k \in \mathbb{N}$. It outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, \mathbf{m} = (m_1, \dots, m_k))$ takes as input a public key pk and a message $\mathbf{m} = (m_1, \dots, m_k) \in \mathbb{G}^k$. It outputs a ciphertext ct.
- $\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \dots, \mathsf{ct}_\ell))$ takes as input a public key pk, a linear function $f : (\mathbb{G}^k)^\ell \to \mathbb{G}^k$ and ℓ ciphertexts $(\mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$. It outputs a new ciphertext $\tilde{\mathsf{ct}}$.
- $\mathsf{Shrink}(\mathsf{pk}, \mathsf{ct})$ takes as input a public key pk and a ciphertext ct. It outputs a new shrunken ciphertext ct'.
- $\mathsf{DecShrink}(\mathsf{sk}, \mathsf{ct})$ takes as input a secret key sk and a shrunken ciphertext ct. It outputs a message $\mathbf{m}$.

For simplicity, we define the algorithm $\mathsf{Eval\&Shrink}(\mathsf{pk}, f, (\mathsf{ct}_1 \dots, \mathsf{ct}_\ell))$ which outputs a ciphertext $\tilde{\mathsf{ct}}$ and is defined as

$$\mathsf{Eval\&Shrink}(\mathsf{pk}, f, (\mathsf{ct}_1 \dots, \mathsf{ct}_\ell)) = \mathsf{Shrink}(\mathsf{pk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \dots, \mathsf{ct}_\ell)))$$

for any linear function f.

We require the following properties from a (circuit-private) packed LHE: Correctness, semantic security, compactness and circuit privacy.

Definition 4.5.2 (Correctness). A packed LHE scheme LHE is said to be correct if for any $\ell \in \mathbb{N}$, any messages $\mathbf{m}_1, \dots, \mathbf{m}_\ell$ and any linear function $f : (\mathbb{G}^k)^\ell \to \mathbb{G}^k$ we have that

$$\Pr\left[\tilde{\mathbf{m}} \leftarrow \mathsf{DecShrink}(\mathsf{sk}, \tilde{\mathsf{ct}}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i) \text{ for } i \in [\ell] \\ \tilde{\mathsf{ct}} \leftarrow \mathsf{Eval\&Shrink}(\mathsf{pk}, , f, (\mathsf{ct}_1 \dots, \mathsf{ct}_\ell)) \end{array}\right] = 1$$

where $\tilde{\mathbf{m}} \leftarrow f(\mathbf{m}_1, \dots, \mathbf{m}_\ell)$.

Definition 4.5.3 (Semantic Security)**.** A packed LHE scheme LHE is said to be semantically secure if for all $\lambda \in \mathbb{N}$, all $k = \mathsf{poly}(\lambda)$ and all adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ we have that

$$\left| \Pr\left[b \leftarrow \mathcal{A}_1(\mathsf{st}, \mathsf{ct}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k) \\ (\mathbf{m}_0, \mathbf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pk}) \\ b \leftarrow\!\!\$ \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Definition 4.5.4 (Compactness)**.** We require that a packed LHE scheme LHE has the following compactness properties:

- For $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k)$, the size of the public key $|\mathsf{pk}|$ is bounded by $k \cdot \mathsf{poly}(\lambda)$.
- For any linear function $f : (\mathbb{G}^k)^\ell \to \mathbb{G}^k$ and any $(\mathbf{m}_1, \ldots, \mathbf{m}_\ell) \in (\mathbb{G}^k)^\ell$ we have that

  $$\lim_{\lambda \to \infty} \inf \frac{|f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)|}{|\mathsf{Eval\&Shrink}(\mathsf{pk}, , f, (\mathsf{ct}_1 \ldots, \mathsf{ct}_\ell))|} \to 1$$

  for sufficiently large k, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k)$ and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i)$ for $i \in [\ell]$. In this case, we say that the scheme has a rate-1.

We also need that the packed LHE scheme fulfills circuit privacy (in the semi-honest case).

Definition 4.5.5 (Circuit Privacy)**.** A packed LHE scheme LHE is said to be circuit-private if for all messages $(\mathbf{m}_1, \ldots, \mathbf{m}_\ell) \in (\mathbb{G}^k)^\ell$ and all linear functions $f : (\mathbb{G}^k)^\ell \to \mathbb{G}^k$, there exists a simulator Sim such that for all adversaries $\mathcal{A}$ we have that

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}, \tilde{\mathsf{ct}}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i) \text{ for } i \in [\ell] \\ \tilde{\mathsf{ct}} \leftarrow \mathsf{Eval\&Shrink}(\mathsf{pk}, , f, (\mathsf{ct}_1 \ldots, \mathsf{ct}_\ell)) \end{array} \right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}, \tilde{\mathsf{ct}}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k) \\ \tilde{\mathsf{ct}} \leftarrow \mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}}) \end{array} \right] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

where $\tilde{\mathbf{m}} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$.

In other words, since Sim does not use f to compute $\tilde{\mathsf{ct}}$, no information about it is leaked from $\tilde{\mathsf{ct}}$ (apart from what is trivially leaked by f).

Encryption of matrices. Above, we defined LHE that supports encryption of vectors $\mathbf{m} \in \mathbb{G}^k$. We can easily extend the definition to support encryption of matrices $\mathbf{M} \in \mathbb{G}^{k \times \alpha}$ for any $\alpha = \mathsf{poly}(\lambda)$: Given a public key pk, encryption $\mathsf{Enc}(\mathsf{pk}, \mathbf{M})$ of $\mathbf{M}$ is defined as

$$\mathsf{Enc}(\mathsf{pk}, \mathbf{M}) = \begin{pmatrix} | & & | \\ \mathsf{Enc}\left(\mathsf{pk}, \mathbf{m}^{(1)}\right) & \ldots & \mathsf{Enc}\left(\mathsf{pk}, \mathbf{m}^{(\alpha)}\right) \\ | & & | \end{pmatrix}$$

where $\mathbf{m}^{(i)}$ is the i-th column of $\mathbf{M}$.

4.5.1 Construction from LWE

Before sketching the scheme, we present the LWE assumption [Rego5].

Definition 4.5.6 (Learning with Errors). Let $n, q \in \mathbb{Z}$. The LWE assumption holds if for any PPT adversary $\mathcal{A}$

$$|\Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e})] - \Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{u})]| \leq \mathsf{negl}(\lambda)$$

for all $m = \mathsf{poly}(n)$, where $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow\!\!\$\ D_{\mathbb{Z},\sigma}^m$ and $\mathbf{u} \leftarrow\!\!\$\ \mathbb{Z}_q^m$.

When we consider $\sigma = \zeta q \geq 2\sqrt{n}$, the LWE problem is at least as hard as solving the approximate shortest independent vector problem to within a factor of $\tilde{\mathcal{O}}(n/\zeta)$ [Rego5].

Shrinking ciphertexts

The work of [BDGM19] shows how to shrink LWE-based ciphertexts of the form $(\mathbf{Ar}, \mathbf{b}_1\mathbf{r} + \lceil q/2 \rfloor m_1, \dots, \mathbf{b}_k\mathbf{r} + \lceil q/2 \rfloor m_k)$ (where $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}^{n \times m}$, $\mathbf{b}_i$ are LWE samples and $\mathbf{r}$ is a short vector). The resulting shrunken ciphertext is composed by $(\mathbf{Ar}, b_1, \dots, b_k)$ where $b_1, \dots, b_k \in \{0,1\}$ and thus the rate tends to 1 when we consider large k.

Before presenting the result of [BDGM19], we first need to define *relaxed correctness* for a standard LHE (as in [BDGM19]). A standard LHE is an LHE where the algorithms Shrink and DecShrink are replaced by a decryption algorithm $\mathbf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$, and it is not required to have rate 1.

Definition 4.5.7 (Relaxed correctness). Let $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a (standard) LHE. We say that LHE is correct with B-noise if

$$\mathbf{Tm} + \mathbf{e} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{Enc}(\mathsf{pk}, \mathbf{m}_1), \dots, \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_\ell))))$$

where $\mathbf{T}$ is an encoding matrix and $\|\mathbf{e}\| \leq B$.

Lemma 4.5.1 ([BDGM19]). *Let* $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a (standard) LHE that is correct with B-noise. Additionally, assume that the ciphertexts of the scheme are of the form* $(\mathbf{c}_1, \mathbf{c}_2)$*, the secret key is of the form* $\mathbf{S} \in \mathbb{Z}_q^{k \times n}$ *and noisy decryption works by computing* $\mathbf{c}_2 - \mathbf{S}\mathbf{c}_1$*. If* $q > 4kB$ *then there exist a correct shrinking algorithm* $(\mathsf{Shrink}_{\mathsf{LWE}}, \mathsf{DecShrink}_{\mathsf{LWE}})$ *for the packed LWE-based LHE scheme.*

Circuit-private LHE from LWE.

We now present the circuit-private LHE from LWE. The scheme is a hybrid between the packed Regev PKE [GPV08] and GSW PKE [GSW13], together with the circuit-privacy technique of [BdMW16].

We present a scheme supporting plaintext space $\{0,1\}^k$. We later briefly explain how we can extend the scheme to any $q = \mathsf{poly}(\lambda)$.

We will need the following ingredients: Let $(\mathsf{Shrink}_{\mathsf{LWE}}, \mathsf{DecShrink}_{\mathsf{LWE}})$ be the pair of algorithms from Lemma 4.5.1. Let $\sigma, \alpha, \beta, q, m, n, t, k$ be polynomials in λ. Let $\mathbf{g}^{-1}_{\mathsf{rnd}}$ be the (randomized) function defined in Section 1 that receives $v \in \mathbb{Z}_p$ as input and outputs $x \leftarrow\!\!\$\ D_{\Lambda_q^{\perp}(\mathbf{g})+\mathbf{g}^{-1}(v),\gamma}$ for some $\gamma = \tilde{\mathcal{O}}(1)$. As defined in Section 1, let $\mathbf{G}_i$ be the matrix with k rows which is zero everywhere except for the i-th row which is equal to $\mathbf{g} = (1, 2, 2^2, \dots, 2^t)$, and let $\bar{\mathbf{G}}_j$ be the matrix with j rows and where every row is equal to $\mathbf{g}$.

$\mathsf{KeyGen}(1^\lambda, k)$:

- Sample $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$, $\mathbf{S} \leftarrow\!\!\$\ \mathbb{Z}_q^{k\times n}$ and $\mathbf{E} \leftarrow\!\!\$\ D_{\mathbb{Z},\sigma}^{k\times m}$. Compute $\mathbf{B} = \mathbf{SA} + \mathbf{E}$.
- Output $\mathsf{pk} = (\mathbf{A}, \mathbf{B})$ and $\mathsf{sk} = \mathbf{S}$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} = (m_1, \dots, m_k) \in \{0,1\}^k)$:

- Parse pk as $(\mathbf{A}, \mathbf{B})$.
- Sample $\mathbf{R} \leftarrow\!\!\$\ D_{\mathbb{Z},\alpha}^{m\times t}$. Compute $\mathbf{C}_1 = \mathbf{AR}$ and $\mathbf{C}_2 = \mathbf{BR} + \sum_{i=1}^{k} m_i \mathbf{G}_i$.
- Output $\mathsf{ct} = (\mathbf{C}_1, \mathbf{C}_2)$.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \dots, \mathsf{ct}_\ell))$

- Parse pk as , f as $f(\mathbf{x}_1, \dots, \mathbf{x}_\ell) = \sum_{i=1}^{\ell} a_i \mathbf{x}_i + \mathbf{b}$, where $\mathbf{a} = (a_1, \dots, a_\ell) \in \mathbb{Z}_2^\ell$, $\mathbf{b} \in \{0,1\}^k$ and ct_i as $(\mathbf{C}_{1,i}, \mathbf{C}_{2,i})$.
- Compute

$$\mathbf{c}_1 = \sum_{j=1}^{\ell} \mathbf{C}_{1,j} \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}\left(\frac{q}{2} a_j\right)^T + \left(\bar{\mathbf{G}}_n - \mathbf{C}_{1,j}\right) \mathbf{g}^{-1}_{\mathsf{rnd}}(0)^T + \mathbf{A}\mathbf{y}_j^T$$

and

$$\mathbf{c}_2 = \sum_{j=1}^{\ell} \left(\mathbf{C}_{2,j} \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}\left(\frac{q}{2} a_j\right)^T + \left(\bar{\mathbf{G}}_k - \mathbf{C}_{2,j}\right) \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}(0)^T + \mathbf{B}\mathbf{y}_j^T \right) + \frac{q}{2}\mathbf{b}$$

where $\mathbf{y}_j \leftarrow\!\!\$\ D_{\mathbb{Z},\beta}^m$.

- Output $\tilde{\mathsf{ct}} = (\mathbf{c}_1, \mathbf{c}_2)$.

$\mathsf{Shrink}(\mathsf{pk}, \mathsf{ct})$: Output $\tilde{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{LWE}}(\mathsf{pk}, \mathsf{ct})$.

$\mathsf{DecShrink}(\mathsf{sk}, \mathsf{ct})$: Output $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{LWE}}(\mathsf{sk}, \mathsf{ct})$.

We now analyze the construction presented above. We start by showing that the scheme is correct.

Lemma 4.5.2 (Correctness). *Let $q = 2q' > 4k(2\alpha\gamma t + \beta)\ell\sigma m\sqrt{k}$ for some $q' \in \mathbb{Z}$. Then the scheme presented above is correct.*

Proof. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k)$ and $\mathsf{ct}_i = (\mathbf{C}_{1,i}, \mathbf{C}_{2,i}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i)$ be well-formed ciphertexts for all $i \in [\ell]$. We have to show that

$$\tilde{\mathbf{m}} \leftarrow \mathsf{DecShrink}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{pk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))))$$

where $\tilde{\mathbf{m}} = \sum_{j=1}^{\ell} a_j \mathbf{m}_j + \mathbf{b} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$.

Let $(\mathbf{c}_1, \mathbf{c}_2) \leftarrow \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$. A routine calculation shows that

$$\mathbf{c}_1 = \mathbf{A}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right)$$

and

$$\mathbf{c}_2 = \mathbf{B}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right) + \frac{q}{2}\left(\sum_{j=1}^{\ell} a_j \mathbf{m}_j + \mathbf{b}\right)$$

where the last equality holds because $2|q$.

We first show that the scheme meets the conditions of Lemma 4.5.1. After computing $\mathbf{c}_2 - \mathbf{S}\mathbf{c}_1$ we obtain

$$\frac{q}{2}\left(\sum_{j=1}^{\ell} a_j \mathbf{m}_j + \mathbf{b}\right) + \mathbf{E}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right).$$

Let $\mathbf{e}' = \mathbf{E}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right)$. By Lemma 1.3.1, each row of $\mathbf{R}_j$ has norm at most $\alpha\sqrt{t}$, the vectors $\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right), \mathbf{g}_{\mathsf{rnd}}^{-1}(0)$ have norm at most $\gamma\sqrt{t}$, $\mathbf{y}_j$ has norm at most $\beta\sqrt{m}$ and each row of $\mathbf{E}$ has norm at most $\sigma\sqrt{m}$. Hence

$$\|\mathbf{e}'\| \leq (2\alpha\gamma t + \beta)\ell\sigma m\sqrt{k}.$$

Since $q > 4k(2\alpha\gamma t + \beta)\ell\sigma m\sqrt{k}$ then we are in the conditions of Lemma 4.5.1. Thus, we conclude that

$$\tilde{\mathbf{m}} \leftarrow \mathsf{DecShrink}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{pk}, (\mathbf{c}_1, \mathbf{c}_2)))$$

where $\tilde{\mathbf{m}} = \sum_{j=1}^{\ell} a_j \mathbf{m}_j + \mathbf{b} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$ and the scheme is correct. □

Semantic security can be established by relying on the smoothing lemma together with the LWE assumption [Reg05, MR04].

Lemma 4.5.3 (Semantic security). *Assume that the LWE assumption holds for $\sigma = \zeta q \geq 2\sqrt{n}$ for some $\zeta \in \mathbb{R}$ and $m = \mathsf{poly}((n+k)\log q)$. Also, let $\alpha \geq \omega(\sqrt{\log m})$. Then the scheme is* *semantically secure.*

Proof (Sketch). In the first hybrid, we replace the public key $(\mathbf{A}, \mathbf{B})$ by $(\mathbf{A}, \mathbf{U})$ for a uniformly chosen $\mathbf{U}$ and this change goes unnoticed by the LWE assumption. Next, we can use the smoothing lemma [Reg05, MR04, GPV08] to replace $(\mathbf{A}, \mathbf{U}, \mathbf{AR}, \mathbf{UR})$ by $(\mathbf{A}, \mathbf{U}, \mathbf{V}_1, \mathbf{V}_2)$ where $\mathbf{V}_1, \mathbf{V}_2$ are uniformly chosen. Finally, we can conclude that the encrypted message is statistically hidden and the result follows. □

Before presenting the proof that the scheme is circuit private, we present a lemma that we will need.

Lemma 4.5.4 ([BdMW16, AR16]). *For any $a \in \mathbb{Z}_q$ and any matrix $\mathbf{E} \in \mathbb{Z}^{k\times m}$, let $r = \tilde{\Theta}(\max_i \|\mathbf{e}_i\| \sqrt{\lambda})$ (where $\mathbf{e}_i$ are the rows of $\mathbf{E}$). Then*

$$\mathbf{E} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(a)^T + \mathbf{y}^T \approx_s \mathbf{f}^T$$

where $\mathbf{y} \leftarrow\$ D^k_{\mathbb{Z},r}$ $\mathbf{f} \leftarrow\$ D^k_{\mathbb{Z},r'}$ and $r' = r\sqrt{1 + \max_i \|\mathbf{e}_i\|^2}$.

Lemma 4.5.5 (Circuit-privacy). *Let $\beta = \tilde{\Theta}(\alpha\sqrt{2\lambda t})$. Then the scheme presented above is circuit private.*

Proof. To prove circuit-private, we show how we can simulate evaluated ciphertexts. We first present the simulator $\mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}})$:

$\mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}})$:

- Sample $\bar{\mathbf{r}} \leftarrow\$ D^m_{\mathbb{Z},\mu}$. Compute $\mathbf{c}_1 = \mathbf{A}\bar{\mathbf{r}}^T$ and $\mathbf{c}_2 = \mathbf{B}\bar{\mathbf{r}}^T + \frac{q}{2}\tilde{\mathbf{m}}$ where $\mu = \beta\sqrt{\ell\left(1 + \left(\alpha\sqrt{t}\right)^2\right)}$ and $\tilde{\mathbf{m}} = f(\mathbf{m}_1, \ldots, \mathbf{m}_t)$.
- Output $\tilde{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{LWE}}(\mathsf{pk}, (\mathbf{c}_1, \mathbf{c}_2))$.

We now prove that the simulated ciphertext is indistinguishable from a evaluated ciphertext. First, note that evaluated ciphertexts are of the form

$$\mathbf{c}_1 = \sum_{j=1}^{\ell} \mathbf{C}_{1,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T - \left(\bar{\mathbf{G}}_n - \mathbf{C}_{1,j}\right) \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{A}\mathbf{y}_j^T$$

and

$$\mathbf{c}_2 = \sum_{j=1}^{\ell} \left(\mathbf{C}_{2,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T + \left(\bar{\mathbf{G}}_k - \mathbf{C}_{2,j}\right) \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{B}\mathbf{y}_j^T\right) + \frac{q}{2}\mathbf{b}.$$

Writing in matrix form, each term of the sum is of the form

$$\begin{aligned}
&\begin{pmatrix}\mathbf{C}_{1,j}\\ \mathbf{C}_{2,j}\end{pmatrix}\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T+\left(\begin{pmatrix}\bar{\mathbf{G}}_n\\ \bar{\mathbf{G}}_k\end{pmatrix}-\begin{pmatrix}\mathbf{C}_{1,j}\\ \mathbf{C}_{2,j}\end{pmatrix}\right)\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T+\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\mathbf{y}_j^T\\
&=\begin{pmatrix}\mathbf{A}\mathbf{R}_j\\ \mathbf{B}\mathbf{R}_j+\sum_{i=1}^k m_{j,i}\mathbf{G}_i\end{pmatrix}\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T+\left(\begin{pmatrix}\bar{\mathbf{G}}_n\\ \bar{\mathbf{G}}_k\end{pmatrix}-\begin{pmatrix}\mathbf{A}\mathbf{R}_j\\ \mathbf{B}\mathbf{R}_j+\sum_{i=1}^k m_{j,i}\mathbf{G}_i\end{pmatrix}\right)\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T+\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\mathbf{y}_j^T\\
&=\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\left(\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T-\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T+\mathbf{y}_j^T\right)+\begin{pmatrix}0\\ a_j\cdot\mathbf{m}_j\end{pmatrix}
\end{aligned}$$

where $\mathbf{m}_j=(m_{j,1},\ldots,m_{j,t})$.

It follows that

$$\begin{aligned}
&\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\left(\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T-\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T+\mathbf{y}_j^T\right)\\
&\approx_s\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\left(\left(\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T+\mathbf{y}_{j,1}^T\right)+\left(-\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T+\mathbf{y}_{j,2}^T\right)\right)\\
&\approx_s\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\left(\mathbf{r}'_{1,j}+\mathbf{r}'_{2,j}\right)\\
&\approx_s\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\mathbf{r}_j'^T
\end{aligned}$$

for $\mathbf{y}_{j,b}\leftarrow\!\!\$\; D_{\mathbb{Z},\beta/\sqrt{2}}$, $\mathbf{r}'_{j,1},\mathbf{r}'_{j,2}\leftarrow\!\!\$\; D^m_{\mathbb{Z},r'/\sqrt{2}}$ and $\mathbf{r}'_j\leftarrow\!\!\$\; D^m_{\mathbb{Z},r'}$ where $r'=\beta\sqrt{1+\left(\alpha\sqrt{t}\right)^2}$. The first and the last steps follow from the fact that the sum of two independent discrete gaussians is statistically close to a discrete gaussian (that is, $\mathbf{y}_j\approx_s\mathbf{y}_{j,1}+\mathbf{y}_{j,2}$ and $\mathbf{r}'_j\approx_s\mathbf{r}'_{j,1}+\mathbf{r}'_{j,2}$). The second step follows from Lemma 4.5.4.

Hence,

$$\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\left(\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T-\mathbf{R}_j\cdot\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T+\mathbf{y}_j^T\right)+\begin{pmatrix}0\\ a_j\cdot\mathbf{m}_j\end{pmatrix}\approx_s\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\mathbf{r}_j'^T+\begin{pmatrix}0\\ a_j\cdot\mathbf{m}_j\end{pmatrix}.$$

From this, we conclude that

$$\begin{aligned}
\begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix} &= \sum_{j=1}^{\ell} \begin{pmatrix} \mathbf{C}_{1,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T - \left(\bar{\mathbf{G}}_n - \mathbf{C}_{1,j}\right)\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{A}\mathbf{y}_j^T \\ \mathbf{C}_{2,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T + \left(\bar{\mathbf{G}}_k - \mathbf{C}_{2,j}\right)\mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{B}\mathbf{y}_j^T \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{q}{2}\mathbf{b} \end{pmatrix} \\
&\approx_s \sum_{j=1}^{\ell} \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} \mathbf{r}_j'^T + \begin{pmatrix} 0 \\ \frac{q}{2}\left(\sum_{j=1}^{\ell} a_j\mathbf{m}_j + \mathbf{b}\right) \end{pmatrix} \\
&\approx_s \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} \bar{\mathbf{r}}^T + \begin{pmatrix} 0 \\ \frac{q}{2} \cdot f(\mathbf{m}_1, \dots, \mathbf{m}_t) \end{pmatrix}
\end{aligned}$$

where $\bar{\mathbf{r}} \leftarrow\!\!\$\; D^m_{\mathbb{Z},\mu}$. The last step follows from the fact that the sum of ℓ independent discrete gaussians with parameter r' (where $r' = \beta\sqrt{1 + \left(\alpha\sqrt{t}\right)^2}$) is statistically indistinguishable from a discrete gaussian with parameter $\sqrt{\ell}r'$. $\square$

Ciphertext rate. It is easy to see that the rate of the ciphertext tends to 1 for large enough k by relying on Lemma 4.5.1. It is also easy to see that the size of the public key $\mathsf{pk} = (\mathbf{A}, \mathbf{B} = \mathbf{SA} + \mathbf{E}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{k\times m}$ is bounded by $k \cdot \mathsf{poly}(\lambda)$.

Larger plaintext space. In the construction presented above, the plaintext space is $\mathbb{Z}_2^k$. The construction can be extended to support plaintext space $\mathbb{Z}_p^k$ for any $p = \mathsf{poly}(\lambda)$ by choosing the LWE modulus q of the form $q = pp'$ where p, p' are co-prime and encoding the encrypted message by q/p.

4.5.2 Construction from DDH

In the following, let $\mathcal{G}$ be a (prime-order) *group generator*, that is, $\mathcal{G}$ is an algorithm that takes as an input a security parameter 1^λ and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is the description of a multiplicative cyclic group, p is the order of the group which is always a prime number unless differently specified, and g is a generator of the group. In the following, we state the decisional version of the Diffie-Hellman (DDH) assumption.

Definition 4.5.8 (Decisional Diffie-Hellman Assumption). Let $(\mathbb{G}, p, g) \leftarrow\!\!\$\; \mathcal{G}(1^\lambda)$. We say that the DDH assumption holds (with respect to $\mathcal{G}$) if for any PPT adversary $\mathcal{A}$

$$\left|\Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), (g^a, g^b, g^{ab}))] - \Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), (g^a, g^b, g^c))]\right| \le \mathsf{negl}(\lambda)$$

where $a, b, c \leftarrow\!\!\$\; \mathbb{Z}_p$.

Shrinking ciphertexts.

We first present how we can shrink DDH-based ciphertexts to achieve rate 1. The shrinking mechanism presented below is a modification of the one presented in [BBD+20] (which is itself based on previous works [BGI16, DGI+19]).

Let $(\mathbb{G}, p, g) \leftarrow\$ \mathcal{G}(1^\lambda)$ and $k \in \mathbb{Z}$. Consider an El Gamal public key of the form $\mathsf{pk} = (g, (h_1, \ldots, h_k) = (g, (g^{x_1}, \ldots, g^{x_k})) \in \mathbb{G}^{k+1}$ for $x_1, \ldots, x_k \leftarrow\$ \mathbb{Z}_p$ (here, $\mathbf{x} = (x_1, \ldots, x_k)$ is the secret key). Consider the following modified El Gamal encryption algorithm where a ciphertext for $\mathbf{m} = (m_1, \ldots, m_k) \in \{0,1\}^k$ is of the form $\mathsf{ct} = (c_1, (c_{2,1}, \ldots, c_{2,k})) \in \mathbb{G}^{k+1}$ where $c_1 = g^r$ and $c_{2,i} = h_i^r g^{\lceil m_i(p/2) \rfloor_\sigma}$. [9] We now show how to compress ciphertexts of this form.

We will need the following ingredients: Let $B, T \in \mathsf{poly}(\lambda)$ and $\mathsf{PRF} = (\mathsf{KeyGen}, \mathsf{Eval})$ be a PRF that maps $g \in \mathbb{G}$ to $\{0,1\}^\tau$ for some $\tau \in \mathbb{Z}$. We also define the function $\mathsf{LEq}_< : \mathbb{G}^2 \to \{0,1\}$ which receives two group elements g_0, g_1 and outputs 1 if $g_0 < g_1$ and 0 otherwise, for some order relation $<$ (e.g. the lexicographic order).

$\mathsf{Shrink}_{\mathsf{DDH}}(\mathsf{pk}, \mathsf{ct})$:

- Parse $\mathsf{pk} = (g, (h_1, \ldots, h_k))$ and $\mathsf{ct} = (c_1, (c_{2,1}, \ldots, c_{2,k}))$. Let $w = g^{\lfloor p/2 \rfloor}$.
- Sample a PRF key $\mathsf{K} \leftarrow\$ \mathsf{PRF.KeyGen}(1^\lambda)$ such that the following conditions are simultaneously satisfied:
  1. For every $i \in [k]$ and $j \in \{-B, \ldots, B\}$ we have that
  $$\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0.$$
  2. For all $i \in [k]$ there exists $\ell \in \{B+1, \ldots, T\}$ such that
  $$\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^\ell) = 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^\ell) = 0.$$
- For every $i \in [k]$, let $\delta_{0,i}, \delta_{1,i} > 0$ be the smallest integer such that
$$\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^{\delta_{0,i}}) = 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^{\delta_{1,i}}) = 0.$$
Let $\alpha_{0,i} = c_{2,i} \cdot g^{\delta_{0,i}}$ and $\alpha_{1,i} = c_{2,i} \cdot w \cdot g^{\delta_{1,i}}$. If $\mathsf{LEq}_<(\alpha_{0,i}, \alpha_{1,i}) = 0$, then set $b_i = 0$. Else, set $b_i = 1$.
- Output $\bar{\mathsf{ct}} = (c_1, \mathsf{K}, (b_1, \ldots, b_k))$.

$\mathsf{DecShrink}_{\mathsf{DDH}}(\mathsf{sk}, \bar{\mathsf{ct}})$:

- Parse $\mathsf{sk} = \mathbf{x} = (x_1, \ldots, x_k)$ and $\bar{\mathsf{ct}} = (c_1, \mathsf{K}, (b_1, \ldots b_k))$. Let $w = g^{\lfloor p/2 \rfloor}$.
- For every $i \in [k]$, compute $\beta_{0,i} = c_1^{x_i}$ and $\beta_{1,i} = c_1^{x_i} \cdot w$.

[9] Note that $\lceil \cdot \rfloor_\sigma$ is defined in section 4.4.

- For every $i \in [k]$, find the smallest integers $\gamma_{0,i}, \gamma_{1,i} > 0$ such that

$$\mathsf{PRF.Eval}(\mathsf{K}, \beta_{0,i} \cdot g^{\gamma_{0,i}}) = 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, \beta_{1,i} \cdot g^{\gamma_{1,i}}) = 0.$$

  Let $\bar{\alpha}_{0,i} = \beta_{0,i} \cdot g^{\gamma_{0,i}}$ and $\bar{\alpha}_{1,i} = \beta_{1,i} \cdot g^{\gamma_{1,i}}$. If $\mathsf{LEq}_<(\bar{\alpha}_{0,i}, \bar{\alpha}_{1,i}) = b_i$, set $m_i = 0$. Else, set $m_i = 1$.

- Output $\mathbf{m} = (m_1, \dots, m_k)$.

Lemma 4.5.6 (Correctness). *Let $B = \mathsf{poly}(\lambda)$ be such that $B > \lambda\sigma + 1$. Then the shrinking procedure presented above is correct.*

Proof. We have to show that $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{DDH}}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{pk}, \mathsf{ct}))$ for $\mathsf{ct} = (c_1, (c_{2,1}, \dots, c_{2,k}))$ where $c_1 = g^r$ and $c_{2,i} = h_i^r g^{\lceil m_i(p/2) \rfloor_\sigma}$ for $i \in [k]$.

For that, we will first show that

$$\begin{array}{c}(\bar{\alpha}_{0,i} = \alpha_{0,i} \wedge \bar{\alpha}_{1,i} = \alpha_{1,i}) \\ \vee \\ (\bar{\alpha}_{0,i} = \alpha_{1,i} \wedge \bar{\alpha}_{1,i} = \alpha_{0,i})\end{array}.$$

We have that $\alpha_{0,i} \neq \alpha_{1,i}$.

The first observation is that $0 \in \{z_0 - (B-1), \dots, z_0 + (B-1)\}$ where $z_0 = \lceil 0 \rfloor_\sigma$ except with negligible probability. This is because $B > \lambda\sigma + 1$ and $|z_0| < \lambda\sigma$ except with negligible probability.[10] Thus $0 \in \{z_0 - B, \dots, z_0 + B\}$

Likewise, $p/2 \in \left[z_{p/2} - (B-1), z_{p/2} + (B-1)\right]$ where $z_{p/2} = \lceil p/2 \rfloor_\sigma$ and thus $\lfloor p/2 \rfloor \in \{z_{p/2} - B, \dots, z_{p/2} + B\}$.

We divide the proof in two cases: Either $m_i = 0$ or $m_i = 1$. We start by analyzing the case where $m_i = 0$.

Case $m_i = 0$. Assume that $m_i = 0$. We first note that $\bar{\alpha}_{0,i} = \beta_{0,i} \cdot g^{\gamma_{0,i}} = c_1^{x_i} \cdot g^{\gamma_{0,i}} = h_i^r \cdot g^{\gamma_{0,i}}$, and $\alpha_{0,i} = h_i^r \cdot g^{z_{0,i} + \delta_{0,i}}$ where $z_{0,i} = \lceil m_i(p/2) \rfloor_\sigma = \lceil 0 \rfloor_\sigma$.We prove that $\bar{\alpha}_{0,i} = \alpha_{0,i}$. To prove this, it is enough to show that $\gamma_{0,i} = z_{0,i} + \delta_{0,i}$. Observe that, if this is not the case, then one of the two cases must be true:

(i) $\gamma_{0,i} < \lceil m_i(p/2) \rfloor_\sigma + \delta_{0,i} = \lceil 0 \rfloor_\sigma + \delta_{0,i}$: If this happens then one of the three cases must hold:

  (a) $\gamma_{0,i} < z_{0,i} - B$: This case cannot hold since $0 \in \{z_{0,i} - B, \dots, z_{0,i} + B\}$ (except with negligible probability) and $\gamma_{0,i} \geq 0$. This implies that $\gamma_{0,i} \geq z_{0,i} - B$, except with negligible probability.

[10]Recall that for a gaussian random variable X centered on 0 and with parameter σ, the probability that $|X| > \lambda\sigma$ is negligible in λ.

(b) $z_{0,i} - B \leq \gamma_{0,i} \leq z_{0,i} + B$: This case cannot hold since it violates condition 1.

(c) $z_{0,i} + B < \gamma_{0,i} < z_{0,i} + \delta_{0,i}$: This case violates condition 2 since $\delta_{0,i} > B$ is the smallest integer that fulfills $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r \cdot g^{z_{0,i}+\delta_{0,i}})$.

(ii) $\gamma_{0,i} > z_{0,i} + \delta_{0,i}$: This case cannot happen as $\gamma_{0,i}$ is the smallest integer (greater than 0) such that $\mathsf{PRF.Eval}(K, h_i^r \cdot g^{\gamma_{0,i}}) = 0$.

Showing that, if $m_i = 0$, then $\bar{\alpha}_{1,i} = \alpha_{1,i}$ follows an identical reasoning. We conclude that, if $m_i = 0$, then $\bar{\alpha}_{0,i} = \alpha_{0,i} \wedge \bar{\alpha}_{1,i} = \alpha_{1,i}$.

Case $m_i = 1$. Now assume that $m_i = 1$. In this case, we show that $\bar{\alpha}_{1,i} = \alpha_{0,i}$ and $\bar{\alpha}_{0,i} = \alpha_{1,i}$.

First, note that $\bar{\alpha}_{1,i} = h_i^r \cdot g^{\lfloor p/2 \rfloor + \gamma_{1,i}}$ and $\alpha_{0,i} = h_i^r \cdot g^{z_{p/2,i}+\delta_{0,i}}$ where $z_{p/2,i} = \lceil m_i(p/2) \rfloor_\sigma = \lceil p/2 \rfloor_\sigma$. We prove that $\bar{\alpha}_{1,i} = \alpha_{0,i}$. To show this, we have to prove that $\lfloor p/2 \rfloor + \gamma_{1,i} = z_{p/2,i} + \delta_{0,i}$. Assume that this is not true, then one of the two cases must happen:

(i) $\lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} + \delta_{0,i}$: If this is the case, then one of the three cases must happen:

(a) $\lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} - B$: This case cannot happen because $\lfloor p/2 \rfloor \in \{z_{p/2,i} - B, \ldots, z_{p/2,i} + B\}$ except with negligible probability and $\gamma_{1,i} > 0$.

(b) $z_{p/2,i} - B < \lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} + B$: This case violates 1 since for any value $j \in \{z_{p/2,i} - B, \ldots, z_{p/2,i} + B\}$, we have that $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r \cdot g^{z_{p/2,i}} \cdot g^j) \neq 0$.

(c) $z_{p/2,i} + B < \lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} + \delta_{0,i}$: This case is also impossible since, by condition 2, $\delta_{0,i}$ is the smallest integer (greater than 0) such that $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r \cdot g^{z_{p/2,i}} \cdot g^{\delta_{0,i}}) = 0$.

(ii) $\lfloor p/2 \rfloor + \gamma_{1,i} > z_{p/2,i} + \delta_{0,i}$: Assume that this is the case. Then $\gamma_{1,i}$ is not the smallest integer greater than 0 such that $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r g^{\lfloor p/2 \rfloor + \gamma_{1,i}}) = 0$.

If $m_i = 1$, showing that $\bar{\alpha}_{0,i} = \alpha_{1,i}$ follows an identical reasoning as above.

Wrapping up. We proved that if $m_i = 0$ then $\alpha_{0,i} = \bar{\alpha}_{0,i}$ and $\alpha_{1,i} = \bar{\alpha}_{1,i}$. On the other hand, if $m_i = 1$, then $\alpha_{0,i} = \bar{\alpha}_{1,i}$ and $\alpha_{1,i} = \bar{\alpha}_{0,i}$. Thus, if the encrypted value is $m_i = 0$

$$b_i = \mathsf{LEq}_<(\alpha_{0,i}, \alpha_{1,i}) = \mathsf{LEq}_<(\bar{\alpha}_{0,i}, \bar{\alpha}_{1,i})$$

and the value output by $\mathsf{DecShrink}_{\mathsf{DDH}}$ is 0. Else if $m_i = 0$

$$b_i = \mathsf{LEq}_<(\alpha_{0,i}, \alpha_{1,i}) \neq \mathsf{LEq}_<(\bar{\alpha}_{0,i}, \bar{\alpha}_{1,i})$$

and the value output by $\mathsf{DecShrink}_{\mathsf{DDH}}$ is 1. □

Lemma 4.5.7 (Runtime). *Let* PRF *be a PRF,* $\tau = \log(8Bk)$ *and* $T = 2^\tau \lambda \log_e(k) + B(1 + 4k)$. *Then, the shrinking algorithm* $\mathsf{Shrink}_{\mathsf{DDH}}$ *described above terminates in polynomial time, except with negligible probability.*

Proof. The analysis of the runtime follows the same reasoning as the analysis of the runtime of the shrinking procedure from [BBD+20].

We have to show that the algorithm $\mathsf{Shrink}_{\mathsf{DDH}}$ is able to find a PRF key K that fulfills both conditions in expected polynomial time, since all other subroutines run in polynomial time. Here, we treat $\mathsf{PRF.Eval}$ as a truly random function. The same analysis is true for the case where $\mathsf{PRF.Eval}$ is a PRF except with negligible probability.

We first lower-bound the probability that a certain PRF key $\mathsf{K} \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$ satisfies condition 1. That is,

$$\begin{aligned}\Pr\left[\forall i \in [k], \forall j \in \{-B, \ldots, B\}, \quad \begin{array}{c}\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \\ \wedge \\ \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0\end{array}\right] &\geq \left(1 - \frac{1}{2^\tau}\right)^{4Bk} \\ &\geq 1 - \frac{4Bk}{2^\tau} \\ &= 1 - \frac{1}{2} = \frac{1}{2}.\end{aligned}$$

Here, the first inequality comes from the fact that the outputs of $\mathsf{PRF.Eval}(K, \cdot)$ are uniform and independent over $\{0,1\}^\tau$ and the second inequality is simply Bernoulli's inequality.

We now upper-bound the probability that condition 2 is not met given that condition 1 happens. Let S be the set of PRF keys for which condition 1 is satisfied. Then

$$\begin{aligned}&\Pr\left[\exists i \in [k] \forall j \in \{B+1, \ldots, T\} : \quad \begin{array}{c}\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \\ \vee \\ \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0\end{array} \middle| K \in S\right] \\ &\leq \sum_{i=1}^{k} \Pr\left[\forall j \in \{B+1, \ldots, T\} : \quad \begin{array}{c}\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \\ \vee \\ \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0\end{array} \middle| K \in S\right] \\ &\leq \sum_{i=1}^{k} \left(1 - \frac{1}{2^\tau}\right)^{T-B-4Bk} \leq \sum_{i=1}^{k} e^{-(T-B-4Bk)/2^\tau} = \sum_{i=1}^{k} e^{-\lambda \log_e(k)} = e^{-\lambda}.\end{aligned}$$

Here, the first inequality is a simple consequence of the union bound and the second inequality follows from observing that K fixes $\mathsf{PRF.Eval}(\mathsf{K}, \cdot)$ on at most $4Bk$ points.

We conclude that, after λ iterations of the protocol, the probability that all the keys do not fulfill both conditions is negligible in λ. $\square$

Ciphertext rate. After applying $\mathsf{Shrink}_{\mathsf{DDH}}$ we obtain a ciphertext composed by $\tilde{\mathsf{ct}} = (c_1, \mathsf{K}, (b_1, \ldots, b_k)) \in \mathbb{G} \times \mathcal{K} \times \{0,1\}^k$. Hence,

$$\frac{|\tilde{\mathsf{ct}}|}{|\mathbf{m}|} = \frac{|c_1| + |\mathsf{K}| + |(b_1, \ldots, b_k)|}{k} = \frac{2\lambda + k}{k} = 1 + \frac{2\lambda}{k}$$

which tends to 1 for large enough k.

Function-private LHE from DDH.

We now present our circuit-private LHE over $\mathbb{Z}_2$ based on DDH.

$\mathsf{KeyGen}(1^\lambda, k)$:

- $(\mathbb{G}, p, g) \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$
- Sample $x_1, \dots, x_k \leftarrow\!\!\$\ \mathbb{Z}_p$. Compute $h_i = g^{x_i}$.
- Output $\mathsf{pk} = (\mathbb{G}, p, g, h_1, \dots, h_k)$ and $\mathsf{sk} = \mathbf{x} = (x_1, \dots, x_k)$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} = (m_1, \dots, m_k))$:

- Parse pk as $(\mathbb{G}, p, g, h_1, \dots, h_k)$.
- Sample $r \leftarrow\!\!\$\ \mathbb{Z}_p$. Compute $c_1 = g^r$ and $c_{2,i} = h_i^r g^{m_i}$ for $i \in [k]$.
- Output $\mathsf{ct} = (c_1, (c_{2,1}, \dots, c_{2,k}))$.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \dots, \mathsf{ct}_\ell))$

- Parse pk as $(\mathbb{G}, p, g, h_1, \dots, h_k)$, f as $f(\mathbf{x}_1, \dots, \mathbf{x}_\ell) = \sum_{i=1}^{\ell} a_i \mathbf{x}_i + \mathbf{b}$ for $\mathbf{a} = (a_1, \dots, a_\ell) \in \mathbb{Z}_2^\ell$ and $\mathbf{b} \in \mathbb{Z}_2^k$ and ct_i as $(c_{1,i}, \mathbf{c}_{2,i})$ where $\mathbf{c}_{2,i} = (c_{2,1,i}, \dots, c_{2,k,i}))$ for $i \in [\ell]$.
- Compute $\bar{\mathsf{ct}} = (\bar{c}_1, (\bar{c}_{2,1}, \dots, \bar{c}_{2,1}))$ where

$$\bar{c}_1 = \prod_{i=1}^{\ell} \left(c_{1,i}^{\lceil a_i \frac{p}{2} \rfloor_\sigma} \cdot (g \cdot c_{1,i}^{-1})^{\lceil 0 \rfloor_\sigma} \right) \cdot g^t$$

  and

$$\bar{\mathbf{c}}_2 = \bigodot_{i=1}^{\ell} \left(\mathbf{c}_{2,i}^{\lceil a_i \frac{p}{2} \rfloor_\sigma} \odot (g \cdot \mathbf{c}_{2,i}^{-1})^{\lceil 0 \rfloor_\sigma} \right) \odot \left(g^{\lceil b_1 \frac{p}{2} \rfloor_\sigma}, \dots, g^{\lceil b_k \frac{p}{2} \rfloor_\sigma} \right) \odot (h_1^t, \dots, h_k^t)$$

  for $t \leftarrow\!\!\$\ \mathbb{Z}_p$ and where $\odot$ denotes the component-wise multiplication.
- Output $\bar{\mathsf{ct}}$.

$\mathsf{Shrink}(\mathsf{pk}, \mathsf{ct})$: Output $\bar{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{DDH}}(\mathsf{pk}, \mathsf{ct})$.

$\mathsf{DecShrink}(\mathsf{sk}, \mathsf{ct})$: Output $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{DDH}}(\mathsf{sk}, \bar{\mathsf{ct}})$.

Correctness and expected polynomial runtime of the LHE described above is guaranteed by Lemma 4.5.6 and Lemma 4.5.7 by setting $B > \lambda(\sigma(\sqrt{2\ell}+1))$. Semantic security of the scheme can be established by a simple reduction to the DDH assumption in a similar way as in many previous works (the reduction is similar to the one that proves that El Gamal is semantically secure). It is also easy to see that the scheme has rate-1 for large enough k.

We now show that the scheme is circuit private. Essentially, circuit privacy can be established by resorting to Lemma 4.4.2.

Lemma 4.5.8 (Circuit-privacy). *The scheme presented above is circuit private.*

Proof. We need to show that we can simulate evaluated ciphertexts. We first present the simulator that receives $\tilde{\mathbf{m}} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$.

$\mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}})$

- Sample $t \leftarrow\$ \mathbb{Z}_q$ and $\alpha_i = \lceil \tilde{m}_i \frac{p}{2} \rfloor_{\sqrt{2\ell+1}\sigma}$.
- Compute $\tilde{\mathsf{ct}} = (\tilde{c}_1, (\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,1}))$ where $\tilde{c}_1 = g^t$ and $\tilde{c}_{2,i} = h_i^t g^{\alpha_i}$. Output $\mathsf{ct}' \leftarrow \mathsf{Shrink}_{\mathsf{DDH}}(\mathsf{pk}, \tilde{\mathsf{ct}})$.

We now show that simulated ciphertexts are statistically indistinguishable from the ones output by Eval.

Let $\mathsf{ct}_i = (g^{r_i}, (h_1^{r_i} g^{m_{1,i}}, \ldots, h_k^{r_i} g^{m_{k,i}})$. The output of Eval is $(\tilde{c}_1, (\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,k}))$ where

$$\tilde{c}_1 = g^{\sum_i^\ell \left(r_i \lceil a_i \frac{p}{2} \rfloor_\sigma - r_i \lceil 0 \rfloor_\sigma\right) + t}$$

and

$$\tilde{c}_{2,j} = h^{\sum_i^\ell \left(r_i \lceil a_i \frac{p}{2} \rfloor_\sigma - r_i \lceil 0 \rfloor_\sigma\right) + t} \cdot g^{\sum_i^\ell \left(m_{j,i} \lceil a_i \frac{p}{2} \rfloor_\sigma + (1 - m_{j,i}) \lceil 0 \rfloor_\sigma\right) + \lceil b_i \frac{p}{2} \rfloor_\sigma}.$$

By Lemma 4.4.2 we have that

$$\sum_{i=1}^{\ell} \left(m_{j,i} \left\lceil a_i \frac{p}{2} \right\rfloor_\sigma + (1 - m_{j,i}) \lceil 0 \rfloor_\sigma \right) + \left\lceil b_i \frac{p}{2} \right\rfloor_\sigma \approx_s \left\lceil \tilde{m}_j \frac{p}{2} \right\rfloor_{\sqrt{2\ell+1}\sigma}$$

where $\tilde{m}_j$ is the j-th coordinate of $f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. Hence,

$$g^{\sum_{i=1}^\ell \left(m_{j,i} \lceil a_i \frac{p}{2} \rfloor_\sigma + (1 - m_{j,i}) \lceil 0 \rfloor_\sigma\right) + \lceil b_i \frac{p}{2} \rfloor_\sigma} \approx_s g^{\lceil f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell) \frac{p}{2} \rfloor_{\sqrt{2\ell+1}\sigma}}.$$

Moreover, since $t \leftarrow\$ \mathbb{Z}_p$ then

$$(g^{z+t}, h^{z+t}) \approx_s (g^t, h^t)$$

for any $z \in \mathbb{Z}_p$. We conclude that

$$(\tilde{c}_1, (\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,k})) \approx_s (g^t, (h_1^t g^{\alpha_1}, \ldots, h_k^t g^{\alpha_k})$$

where $\alpha_i = \lceil \tilde{m}_i \frac{p}{2} \rfloor_{\sqrt{2\ell+1}\sigma}$. □

Larger plaintext space. As in the LWE case, in the construction presented above, the plaintext space is $\mathbb{Z}_2^k$. Both the shrinking algorithm and the function-private LHE schemes can be extended to support plaintext space $\mathbb{Z}_q^k$ where $q = \mathsf{poly}(\lambda)$ and $q = 2^\nu$ for some $\nu \in \mathbb{Z}$ (the constrain of q being a power of 2 comes from Lemma 4.4.2)

4.5.3 Construction from QR

The scheme presented in this section is the packed version of the scheme from [BG10] together with the shrinking technique from [DGI$^+$19].

In the following, let N is a Blum integer if $N = p \cdot q$ for some primes p and q such that $p \pmod 4 = q \pmod 4 = 3$. Moreover, we say p and q are safe primes if $p = 2p' + 1$ and $q = 2q' + 1$ for some prime numbers p', q'. We denote by $\mathbb{J}_N$ the multiplicative group of the elements in $\mathbb{Z}_N^*$ with Jacobi symbol $+1$ and by $\mathbb{QR}_N$ the multiplicative group of quadratic residues modulo N with generator g. Note that $\mathbb{QR}_N$ is a subgroup of $\mathbb{J}_N$ and they have order $\frac{\phi(N)}{4}$ and $\frac{\phi(N)}{2}$, respectively, where $\phi(\cdot)$ is Euler's totient function. It is useful to write $\mathbb{J}_N \simeq \mathbb{H} \times \mathbb{QR}_N$, where $\mathbb{H}$ is the multiplicative group $(\pm 1, \cdot)$ of order 2. Note that if N is a Blum integer then $\gcd\left(2, \frac{\phi(N)}{4}\right) = 1$ and $-1 \in \mathbb{J}_N \setminus \mathbb{QR}_N$. We recall the quadratic residuosity (QR) assumption [GM82].

Definition 4.5.9 (Quadratic Residuosity Assumption). Let N be a uniformly sampled Blum integer and let $\mathbb{QR}_N$ be the multiplicative group of quadratic residues modulo N with generator g. We say the QR assumption holds with respect to $\mathbb{QR}_N$ if for any PPT adversary $\mathcal{A}$

$$|\Pr[1 \leftarrow \mathcal{A}(N, g, a)] - \Pr[1 \leftarrow \mathcal{A}(N, g, (-1) \cdot a)]| \leq \mathsf{negl}(\lambda)$$

where $a \leftarrow\!\!\$\ \mathbb{QR}_N$.

Shrinking ciphertexts.

We recall the shrinking mechanism of [DGI$^+$19]. Let a (packed) ciphertext $\mathsf{ct} = (g^r, (-1)^{b_1} h_1^r, \ldots, (-1)^{b_k} h_k^r) = (c_1, c_{2,1}, \ldots, c_{2,k})$ and let $<$ be an order over $\mathbb{J}_N$ (e.g., the lexicographic order). The shrinking mechanism of [DGI$^+$19] simply outputs 0 if $c_{2,i} < -c_{2,i}$ and outputs 1 otherwise.

Lemma 4.5.9 ([DGI$^+$19]). *There exists a correct shrinking procedure* $\mathsf{Shrink}_{\mathsf{QR}}$, $\mathsf{DecShrink}_{\mathsf{QR}}$ *for the packed QR-based PKE.*

Circuit-private LHE from QR.

We now present the scheme which is essentially the same as the one from [BG10] together with the shrinking technique of Lemma 4.5.9.

In the following, let $(\mathsf{Shrink}_{\mathsf{QR}}, \mathsf{DecShrink}_{\mathsf{QR}})$ be the pair of algorithms from Lemma 4.5.9.

$\mathsf{KeyGen}(1^\lambda, k)$:

- Choose two safe primes $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are primes and compute $N = pq$. Choose a generator g of $\mathbb{QR}_N$.
- Sample $\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}^k_{\varphi(N)/2}$ and compute $\mathbf{h} = g^{\mathbf{s}}$.
- Output $\mathsf{pk} = (N, g, \mathbf{h})$ and $\mathsf{sk} = \mathbf{s}$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} = (m_1, \dots, m_k))$:

- Parse pk as $(N, g, \mathbf{h} = (h_1, \dots, h_k))$.
- Sample $r \leftarrow\!\!\$\ \mathbb{Z}_{(N-1)/2}$. Compute $c_1 = g^r$ and $c_{2,i} = (-1)^{m_i} h_i^r$ for $i \in [k]$.
- Output $\mathsf{ct} = (c_1, \mathbf{c}_2 = (c_{2,1}, \dots, c_{2,k}))$.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \dots, \mathsf{ct}_\ell))$

- Parse pk as $(N, g, \mathbf{h} = (h_1, \dots, h_k))$, f as $f(\mathbf{x}_1, \dots, \mathbf{x}_\ell) = \sum_{j=1}^{\ell} a_j \mathbf{x}_j + \mathbf{b}$, where $a_1, \dots, a_\ell \in \mathbb{Z}_2$, $\mathbf{b} \in \mathbb{Z}_2^k$ and ct_j as $(c_{1,j}, \mathbf{c}_{2,j} = (c_{2,1,j}, \dots, c_{2,k,j}))$.
- Compute $\tilde{\mathsf{ct}} = (\tilde{c}_1, \tilde{\mathbf{c}}_2 = (\tilde{c}_{2,1}, \dots, \tilde{c}_{2,k}))$ where $\tilde{c}_1 = g^t \prod_{j=1}^{\ell} c_{1,j}^{a_j}$ and $\tilde{c}_{2,i} = h_i^t \cdot (-1)^{b_i} \cdot \prod_{j=1}^{\ell} c_{2,i,j}^{a_j}$ where $t \leftarrow\!\!\$\ \mathbb{Z}_{(N-1)/2}$. Output $\tilde{\mathsf{ct}}$.

$\mathsf{Shrink}(\mathsf{pk}, \mathsf{ct})$: Output $\tilde{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{QR}}(\mathsf{pk}, \mathsf{ct})$.

$\mathsf{DecShrink}(\mathsf{sk}, \mathsf{ct})$: Output $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{QR}}(\mathsf{sk}, \mathsf{ct})$.

It is easy to see that correctness and compactness hold due to Lemma 4.5.9. Semantic security also follows easily from the QR assumption.

To see that the scheme is circuit private, note that $g^{(\sum_j r_j a_j)+t} \approx_s g^t$ for a uniformly chosen $t \leftarrow\!\!\$\ \mathbb{Z}_{(N-1)/2}$ (this holds since the uniform distribution over $\mathbb{Z}_{(N-1)/2}$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_{\varphi(N)/2}$). Similarly, we have that $h_i^{(\sum_j r a_j)+t} \approx_s h_i^t$. Thus,

$$\left(g^{(\sum_j r_j a_j)+t}, (-1)^{f(\mathbf{m})} h^{(\sum_j r_j a_j)+t}\right) \approx_s \left(g^t, (-1)^{f(\mathbf{m})} h^t\right)$$

and, thus, the distributions of an evaluated ciphertext and a fresh ciphertext are statistically indistinguishable.

4.6 Co-Private Information Retrieval

In this section, we present a new cryptographic primitive that we call *co-PIR*. In a co-PIR scheme, a receiver (with input a set of indices S) and a sender (with no input) interact such that, in the end, the sender obtains a string $\mathbf{y} \in \mathbb{Z}_q^m$ and the receiver obtains $\mathbf{y}_{-S}$ (all positions of y except for the indices in S).

In terms of security, we require that the sender learns nothing about S, whereas the string $\mathbf{y}_S$ looks pseudorandom to the receiver. In terms of efficiency, we require that the total communication of the protocol scales only with $|S|\mathsf{poly}(\lambda)\mathsf{polylog}(m)$ (that is, it scales only poly-logarithmically with m).

4.6.1 Definition

We start by defining Co-PIR and presenting its security properties.

Definition 4.6.1 (Co-PIR). A (two-round) Co-PIR scheme CoPIR over $\mathbb{Z}_q$ is parametrized by an integer m where $m = \mathsf{poly}[\lambda]$, and is composed by a tuple of algorithms $(\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ such that

- $\mathsf{Query}(1^\lambda, S)$ takes as input a set of indices $S \subseteq [m]$. It outputs a message copir_1 and a private state st.
- $\mathsf{Send}(\mathsf{copir}_1)$ takes as input a first message copir_1. It outputs a second message copir_2 and a string $\mathbf{y} \in \mathbb{Z}_q^m$.
- $\mathsf{Dec}(\mathsf{copir}_2, \mathsf{st})$ takes as input a second message copir_2 and a state st. It outputs a string $\tilde{\mathbf{y}} \in \mathbb{Z}_q^m$.

Definition 4.6.2 (Correctness). A Co-PIR scheme CoPIR is said to be correct if for any $m = \mathsf{poly}(\lambda)$ and $S \subseteq [m]$ we have that

$$\Pr\left[\mathbf{y}_{[m]\setminus S} = \tilde{\mathbf{y}}_{[m]\setminus S} : \begin{array}{c} (\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S) \\ (\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1) \\ \tilde{\mathbf{y}} \leftarrow \mathsf{Retrieve}(\mathsf{copir}_2, \mathsf{st}) \end{array}\right] = 1.$$

In other words, the strings $\mathbf{y}$ and $\tilde{\mathbf{y}}$ match for every coordinate $i \in [m] \setminus S$.

In terms of security, we require two properties: receiver security and sender security.

Definition 4.6.3 (Receiver security). A Co-PIR scheme CoPIR is said to be receiver secure if for all $m = \mathsf{poly}(\lambda)$, any subsets $S_1, S_2 \subseteq [m]$ we have that for any adversary $\mathcal{A}$

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{copir}_1) : (\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S_1)\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{copir}_1) : (\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S_2)\right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

Definition 4.6.4 (Sender security). A Co-PIR scheme CoPIR is said to be sender secure if for any $m = \mathsf{poly}(\lambda)$, any subset $S \subseteq [m]$ we have that for all adversaries $\mathcal{A}$

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{st}, \mathsf{copir}_2, \mathbf{y}_S) : \begin{array}{c} (\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S) \\ (\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1, \mathbf{x}) \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{st}, \mathsf{copir}_2, \mathbf{y}'_S) : \begin{array}{c} (\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S) \\ (\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1, \mathbf{x}) \\ \mathbf{y}'_S \leftarrow\!\!\$\; \mathbb{Z}_q^{|S|} \end{array}\right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

Definition 4.6.5 (Compactness). A Co-PIR scheme CoPIR is said to be compact if $|\mathsf{copir}_1|, |\mathsf{copir}_2| = |S| \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$ for any $S \subseteq [m]$ where $(\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S)$ and $(\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1)$. In other words, the communication complexity depends only on poly-logarithmically in m.

4.6.2 Co-PIR from Distributed GGM-PPRF Correlation

We now present a scheme for Co-PIR from the distributed GGM-PPRF correlation which is proposed by Boyle et al. [BCG+19a]. For the sake of simplicity, we present the scheme for $q = 2$. Let $\mathsf{PPRF}_{\mathsf{GGM}} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Punct}, \mathsf{EvalPunct})$ be a GGM puncturable PRF which maps from $[m]$ to $\{0,1\}$ and let $\mathsf{PPRF\text{-}GGM} = (\mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ be a distributed GGM-PPRF correlation scheme.

$\mathsf{Query}(1^\lambda, S)$:

- Parse $S = \{a_1, \dots, a_t\} \subseteq [m]^t$ where $t = |S|$.
- For $j \in [t]$ compute $(\mathsf{pprf\text{-}ggm}_{1,j}, \mathsf{state}_j) \leftarrow \mathsf{PPRF\text{-}GGM.R}_1(\bar{a}_j)$.
- Output $\mathsf{copir}_1 = \{\mathsf{pprf\text{-}ggm}_{1,j}\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.

$\mathsf{Send}(\mathsf{copir}_1)$:

- Parse $\mathsf{copir}_1 = \{\mathsf{pprf\text{-}ggm}_{1,j}\}_{j\in[t]}$.
- For $j \in [t]$ compute $\mathsf{K}_j \leftarrow \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{KeyGen}(1^\lambda)$ and $\mathbf{z}_j \leftarrow \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_j, *)$.
- For $j \in [t]$ compute $\mathsf{pprf\text{-}ggm}_{2,j} \leftarrow \mathsf{PPRF\text{-}GGM.S}(0, \mathsf{K}_j, \mathsf{pprf\text{-}ggm}_1)$.
- Output $\mathsf{copir}_2 = \{\mathsf{pprf\text{-}ggm}_{2,j}\}_{j\in[t]}$ and $\mathbf{y} = \sum_{i=1}^t \mathbf{z}_j$

$\mathsf{Dec}(\mathsf{copir}_2, \mathsf{st})$:

- Parse $\mathsf{copir}_2 = \{\mathsf{pprf\text{-}ggm}_{2,j}\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.
- For $j \in [t]$ compute $\bar{\mathsf{K}}_j \leftarrow \mathsf{PPRF\text{-}GGM.R}_2(\mathsf{state}_j, \mathsf{pprf\text{-}ggm}_{2,j})$.
- For $i \in [m] \setminus S$, set $\tilde{y}_i = \sum_{i=1}^t \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\bar{\mathsf{K}}_j, i)$. For $i \in S$, set $\tilde{y}_i = 0$. Output $\tilde{\mathbf{y}} = (\tilde{y}_1 \dots, \tilde{y}_m)$.

We now analyze the scheme presented above starting with correctness.

Lemma 4.6.1 (Correctness). *Assume that* $\mathsf{PPRF\text{-}GGM}$ *and* PPRF *are correct. Then the scheme presented above is correct*

Proof. We have to prove that $\tilde{\mathbf{y}}_{[m]\setminus S} = \mathbf{y}_{[m]\setminus S}$. Let $\mathbf{y} = (y_1, \dots, y_m) \in \{0,1\}^m$. Note that $y_i = \sum_{j=1}^t \mathsf{PPRF.Eval}(\mathsf{K}_j, i)$.

First, by the correctness of the underlying distributed GGM-PPRF correlation scheme, $\bar{\mathsf{K}}_j \leftarrow \mathsf{PPRF.Punct}(a_j)$ for all $j \in [t]$ and $a_j \in S$. Also,

$$\tilde{y}_i = \sum_{j=1}^{t} \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\bar{\mathsf{K}}_j, i)$$

for all $i \in [m] \backslash S$. By the correctness of the PPRF, $\mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\bar{\mathsf{K}}_j, i) = \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\mathsf{K}_j, i)$ for all $i \in [m] \setminus S$. Then $\tilde{y}_i = y_i$ for all $i \in [m] \setminus S$. $\square$

Lemma 4.6.2 (Receiver security). *Assume that* $\mathsf{PPRF\text{-}GGM}$ *implements* $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$. *Then the scheme presented above is receiver secure.*

The proof follows directly from the receiver security of $\mathsf{PPRF\text{-}GGM}$.

Lemma 4.6.3 (Sender security). *Assume that* $\mathsf{PPRF\text{-}GGM}$ *implements* $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ *and* PPRF *is a pseudorandom PPRF. Then the scheme presented above is sender secure.*

Proof. Let $\mathsf{Sim}_{\mathsf{PPRF}}\text{-}\mathsf{GGM}$ be the simulator of $\mathsf{PPRF\text{-}GGM}$ for sender security. The proof of security follows the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This is the real protocol.

For all $j \in [t]$ consider the following sub-hybrids.

Hybrid $\mathcal{H}_{1,j}$. In this hybrid, we replace $\mathsf{pprf\text{-}ggm}_{2,j}$ by the message generated by $\mathsf{Sim}_{\mathsf{PPRF}}\text{-}\mathsf{GGM}$. Indistinguishability of hybrids follows from the sender security of $\mathsf{PPRF\text{-}GGM}$.

Hybrid $\mathcal{H}_{2,j}$. In this hybrid, we replace $\mathsf{PRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_j, a_j)$ by a uniform bit $u_j \leftarrow\!\!\$ \{0,1\}$. Indistinguishability of hybrids follows from the pseudorandomness of PPRF.

Hybrid $\mathcal{H}_{3,j}$. In this hybrid, we replace y_{a_j} by $v_j \leftarrow\!\!\$ \{0,1\}$. Statistical indistinguishability follows because

$$y_{a_j} = \sum_{i=1}^{t} \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_i, a_j) = \sum_{i=1, i\neq j}^{t} \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_i, a_j) + u_j \approx_s v_j.$$

Finally, note that in hybrid $\mathcal{H}_{3,t}$ the string $\mathbf{y}_S$ is uniformly random to the receiver and we conclude the proof $\square$

Compactness. To conclude, we analyze compactness of the scheme. Assuming that the distributed GGM-PPRF correlation scheme has polynomial communication complexity in $|a_j| = \log m$ and in λ, and $|S| = t$, we conclude that the receiver's and the sender's message are of size $t \cdot \mathsf{poly}(\lambda) \cdot \mathsf{polylog}(m)$.

Extending to Co-PIR over any $\mathbb{Z}_q$. The scheme can be easily extended to any q by taking a PPRF that maps $x \in [m]$ to $\mathbb{Z}_q$. It is easy to see that the resulting scheme has total communication complexity of $t \cdot \mathsf{poly}(\lambda) \cdot \mathsf{polylog}(m, q)$.

Hardness assumptions for Co-PIR. Since the distributed GGM-PPRF correlation scheme and the GGM-PPRF can be based on LWE, DDH or QR assumptions (using only black-box techniques), then the Co-PIR scheme presented above can also be based on these assumptions. Moreover, the resulting scheme uses only black-box techniques.

4.6.3 Co-PIR from PPRF and PIR

The construction for Co-PIR from Section 4.6 uses a distributed GGM-PPRF correlation scheme which can be built from a GGM-PPRF and an OT. In this section, we present a construction for Co-PIR from any PPRF (not necessarily the GGM-PPRF) and a PIR in a black-box way.

The Protocol

For the sake of simplicity, we present the scheme for $q = 2$.

For our Co-PIR construction, we will need the following ingredients: Let $\mathsf{PIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ be a PIR scheme with poly-logarithmic communication complexity and sender privacy and let $\mathsf{PPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Punct}, \mathsf{EvalPunct})$ be a puncturable PRF which maps from $[m]$ to $\{0, 1\}$. We use the notation $\mathsf{PPRF.Eval}(\mathsf{K}, *)$ to denote the vector $(\mathsf{PPRF.Eval}(\mathsf{K}, 1), \ldots, \mathsf{PPRF.Eval}(\mathsf{K}, m)) \in \{0, 1\}^m$.

$\mathsf{Query}(1^\lambda, S)$:

- Parse $S = \{a_1, \ldots, a_t\}$ where $t = |S|$.
- For $j \in [t]$ compute $(\mathsf{q}_j, \mathsf{state}_j) \leftarrow \mathsf{PIR.Query}(a_j)$.
- Output $\mathsf{copir}_1 = \{\mathsf{q}_j\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.

$\mathsf{Send}(\mathsf{copir}_1)$:

- Parse $\mathsf{copir}_1 = \{\mathsf{q}_j\}_{j\in[t]}$.
- For $j \in [t]$ compute $\mathsf{K}_j \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$ and $\mathbf{z}_j \leftarrow \mathsf{PPRF.Eval}(\mathsf{K}_j, *)$.
- For $i \in [j]$ and $\ell \in [m]$, set $\dot{\mathsf{K}}_{j,\ell} \leftarrow \mathsf{PPRF.Punct}(\mathsf{K}_j, \ell)$.
- For $j \in [t]$ set $\mathbf{DB}_j = (\dot{\mathsf{K}}_{j,1}, \ldots, \dot{\mathsf{K}}_{j,m})$. Compute $\mathsf{r}_j \leftarrow \mathsf{PIR.Send}(\mathbf{DB}_j, \mathsf{q}_j)$.
- Output $\mathsf{copir}_2 = \{\mathsf{r}_j\}_{j\in[t]}$ and $\mathbf{y} = \sum_{i=1}^{t} \mathbf{z}_j$

$\mathsf{Dec}(\mathsf{copir}_2, \mathsf{st})$:

- Parse $\mathsf{copir}_2 = \{\mathsf{r}_j\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.
- For $j \in [t]$ compute $\bar{\mathsf{K}}_j \leftarrow \mathsf{PIR.Retrieve}(\mathsf{r}_j, \mathsf{state}_j)$.
- For $i \in [k] \setminus S$, set $\tilde{y}_i = \sum_{i=1}^{t} \mathsf{PPRF.EvalPunct}(\bar{\mathsf{K}}_j, i)$. For $i \in S$, set $\tilde{y}_i = 0$. Output $\tilde{\mathbf{y}} = (\tilde{y}_1 \dots, \tilde{y}_m)$.

Analysis

We now analyze the scheme presented above starting with correctness.

Lemma 4.6.4 (Correctness). *Assume that* PIR *and* PPRF *are correct. Then the scheme presented above is correct*

Proof. We have to prove that $\tilde{\mathbf{y}}_{[k]\setminus S} = \mathbf{y}_{[k]\setminus S}$. Let $\mathbf{y} = (y_1, \dots, y_m) \in \{0,1\}^m$. Note that $y_i = \sum_{j=1}^{t} \mathsf{PPRF.Eval}(\mathsf{K}_j, i)$.

First, by the correctness of the underlying PIR scheme, $\bar{\mathsf{K}}_j = \dot{\mathsf{K}}_{j,a_j}$ for all $j \in [t]$ and $a_j \in S$. Also,

$$\tilde{y}_i = \sum_{j=1}^{t} \mathsf{PPRF.EvalPunct}(\bar{\mathsf{K}}_j, i) = \sum_{i=1}^{t} \mathsf{PPRF.EvalPunct}(\dot{\mathsf{K}}_{j,a_j}, i)$$

for all $i \in [k] \setminus S$. By the correctness of the PPRF, $\mathsf{PPRF.EvalPunct}(\dot{\mathsf{K}}_{j,a_j}, i) = \mathsf{PPRF.EvalPunct}(\mathsf{K}_j, i)$ for all $i \in [k] \setminus S$. Then $\tilde{y}_i = y_i$ for all $i \in [k] \setminus S$. $\square$

Lemma 4.6.5 (Receiver security). *Assume that* PIR *is user secure. Then the scheme presented above is receiver secure.*

The proof follows from a simple reduction from the receiver security of CoPIR to user security of PIR.

Lemma 4.6.6 (Sender security). *Assume that* PIR *is sender secure and* PPRF *is a pseudorandom PPRF. Then the scheme presented above is sender secure.*

Proof. The proof of security follows the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This is the real protocol.

For all $j \in [t]$ consider the following sub-hybrids.

Hybrid $\mathcal{H}_{1,j}$. In this hybrid, we replace DB_j by $\overline{DB}_j$ which is 0 everywhere but its a_j-th coordinate is equal to DB_{j,a_j}. Indistinguishability of hybrids follows from the sender security of PIR.

Hybrid $\mathcal{H}_{2,j}$. In this hybrid, we replace $\mathsf{PRF.Eval}(\mathsf{K}_j, a_j)$ by a uniform bit $u_j \leftarrow\!\!\$ \{0,1\}$. The indistinguishability of hybrids follows from the pseudorandomness of PPRF.

Hybrid $\mathcal{H}_{3,j}$. In this hybrid, we replace y_{a_j} by $v_j \leftarrow\$ \{0,1\}$. Statistical indistinguishability follows because

$$y_{a_j} = \sum_{i=1}^{t} \mathsf{PPRF.Eval}(\mathsf{K}_i, a_j) = \sum_{i=1, i\neq j}^{t} \mathsf{PPRF.Eval}(\mathsf{K}_i, a_j) + u_j \approx_s v_j.$$

Finally, note that in hybrid $\mathcal{H}_{3,t}$ the string $\mathbf{y}_S$ is uniformly random to the receiver and we conclude the proof $\square$

Compactness. To conclude, we analyze the compactness of the scheme. Assuming that the PIR scheme has poly-logarithmic communication complexity and $|S| = t$, we conclude that the receiver's and the sender's message are of size $t \cdot \mathsf{poly}(\lambda) \cdot \mathsf{polylog}(m)$.

4.7 Oblivious Transfer with Overall Rate 1

We will now provide our construction of an oblivious transfer protocol with an overall rate-1.

Ingredients. We will make use of the following ingredients.

- A packed linearly homomorphic encryption scheme $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Shrink}, \mathsf{DecShrink})$ with plaintext space $\{0,1\}^\ell$ and a post homomorphism shrinking procedure Shrink which converts ciphertexts into a rate 1 representation.[11]
- The binary $\mathsf{LPN}(n, m, \rho)$ problem with dimension $n = \mathsf{poly}(\lambda)$, $m = n \cdot \ell \cdot \mathsf{poly}(\lambda)$ samples and slightly sub-constant noise-rate $\rho = m^{1-\varepsilon}$.
- A 2-round PIR scheme $\mathsf{PIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ with poly-logarithmic communication complexity and sender privacy.
- A 2-round Co-PIR scheme $\mathsf{CoPIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ over $\mathbb{Z}_2$ parametrized by m.

Additional Notation. Furthermore, to declutter notation we define the following embedding functions.

$\mathsf{RowMatrix}(\ell, n, \mathbf{v}_1, \ldots, \mathbf{v}_\ell)$: Takes row-vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \{0,1\}^n$ and outputs a matrix

$$\mathbf{V} = \begin{pmatrix} - & \mathbf{v}_1 & - \\ & \vdots & \\ - & \mathbf{v}_\ell & - \end{pmatrix},$$

i.e. for every $i \in [\ell]$ the i-th row of $\mathbf{V}$ is the row-vector $\mathbf{v}_i$.

[11] Recall that we use the notation $\mathsf{Eval\&Shrink}$ to denote the composition of algorithms Eval and Shrink.

$\mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{v})$**:** Takes a row-vector $\mathbf{v} \in \{0,1\}^n$ and outputs a matrix

$$\mathbf{V} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \\ — & \mathbf{v} & — \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{pmatrix},$$

i.e. the i-th row of $\mathbf{V}$ is $\mathbf{v}$, but $\mathbf{V}$ is 0 everywhere else.

$\mathsf{Diag}(n, \mathbf{v})$**:** Takes a vector $\mathbf{v} = (v_1, \dots, v_n) \in \{0,1\}^n$ and outputs a matrix

$$\mathbf{D} = \begin{pmatrix} v_1 & & 0 \\ & \ddots & \\ 0 & & v_n \end{pmatrix},$$

i.e. $\mathbf{D} \in \{0,1\}^{n \times n}$ is a diagonal matrix with the components of $\mathbf{v}$ on its diagonal.

We observe the following:

- For any $\mathbf{v}_1, \dots, \mathbf{v}_\ell \in \{0,1\}^n$ it holds that

$$\mathsf{RowMatrix}(\ell, n, \mathbf{v}_1, \dots, \mathbf{v}_\ell) = \sum_{i=1}^{\ell} \mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{v}_i).$$

- For $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ it holds that

$$\mathbf{x} \cdot \mathsf{Diag}(n, \mathbf{y}) = \mathbf{x} \odot \mathbf{y},$$

  where $\odot$ denotes component-wise multiplication.

4.7.1 The Protocol

The protocol $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ is given as follows.

$\mathsf{OTR}(\mathbf{b} \in \{0,1\}^{m\ell})$:

- Parse $\mathbf{b} = (\mathbf{b}_1, \dots, \mathbf{b}_\ell)$, where the $\mathbf{b}_i \in \{0,1\}^m$ are blocks of size m.
- Choose $\mathbf{A} \leftarrow\!\!\$\ \{0,1\}^{n \times m}$ uniformly at random and compute a pair of public and secret key $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, \ell)$.

- For all $i \in [\ell]$, choose $\mathbf{s}_i \leftarrow\$ \{0,1\}^n$, and $\mathbf{e}_i \leftarrow\$ \chi_{m,t}$, compute $\mathbf{c}_i \leftarrow \mathbf{s}_i\mathbf{A} + \mathbf{e}_i + \mathbf{b}_i$, and set $\mathbf{S}_i \leftarrow \mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{s}_i)$. Compute a matrix-ciphertext $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_i)$.
- For all $i \in [\ell]$ set $J_i = \mathsf{Supp}(\mathbf{e}_i)$ to be the support of $\mathbf{e}_i$. Compute $(\mathsf{copir}_{1,i}, \mathsf{st}_i) \leftarrow \mathsf{CoPIR.Query}(J_i)$. Additionally, for $j \in [t]$ compute $(\mathsf{q}_{i,j}, \hat{\mathsf{st}}_{i,j}) = \mathsf{PIR.Query}(J_i[j])$.
- Output $\mathsf{ot}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}_{1,i}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$ and $\mathsf{st} = (\mathsf{sk}, \{\mathsf{st}_i, J_i\}_{i\in[\ell]}, \{\hat{\mathsf{st}}_{i,j}\}_{i\in[\ell],j\in[t]]})$.

$\mathsf{OTS}((\mathbf{m}_0, \mathbf{m}_1) \in (\{0,1\}^{m\ell})^2, \mathsf{ot}_1)$:

- Parse $\mathbf{m}_0 = (\mathbf{m}_{0,1}, \ldots, \mathbf{m}_{0,\ell})$ and $\mathbf{m}_1 = (\mathbf{m}_{1,1}, \ldots, \mathbf{m}_{1,\ell})$, where each $\mathbf{m}_{b,i} = (m_{b,i,1}, \ldots, m_{b,i,m}) \in \{0,1\}^m$. Parse $\mathsf{ot}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}_{1,i}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$.
- For $i \in [\ell]$ $(\mathbf{y}_i, \mathsf{copir}_{2,i}) \leftarrow \mathsf{CoPIR.Send}(\mathsf{copir}_{1,i})$ where $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,m})$. Set $\mathbf{z}_i = \mathbf{m}_{0,i} + \mathbf{y}_i$.
- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$.
- For all $i \in [\ell]$ set $\mathbf{C}_i = \mathsf{SingleRowMatrix}(\ell, m, i, \mathbf{c}_i)$ and $\mathbf{D}_i = \mathsf{Diag}(m, \mathbf{m}_{1,i} - \mathbf{m}_{0,i})$.
- Define the $\mathbb{Z}_2$-linear function $f\colon (\{0,1\}^{\ell\times n})^\ell \to \{0,1\}^{\ell\times m}$ via

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_\ell) = \left(\sum_{i=1}^{\ell}(-\mathbf{X}_i\mathbf{A} + \mathbf{C}_i)\cdot\mathbf{D}_i\right) + \mathbf{Z}.$$

- Compute $\tilde{\mathsf{ct}} \leftarrow \mathsf{LHE.Eval\&Shrink}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
- For $i \in [\ell]$ set $DB_i = (y_{i,1} + (m_{1,i,1} - m_{0,i,1}), \ldots, y_{i,m} + (m_{1,i,m} - m_{0,i,m}))$. For all $j \in [t]$ compute $\mathsf{r}_{i,j} \leftarrow \mathsf{PIR.Send}(DB_i, \mathsf{q}_{i,j})$.
- Output $\mathsf{ot}_2 = \big(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$.

$\mathsf{OTD}(\mathsf{ot}_2, \mathsf{st})$:

- Parse $\mathsf{ot}_2 = \big(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$ and $\mathsf{st} = (\mathsf{sk}, \{\mathsf{st}_i, J_i\}_{i\in[\ell]}, \{\hat{\mathsf{st}}_{i,j}\}_{i\in[\ell],j\in[t]]})$.
- For all $i \in [\ell]$ compute $\tilde{\mathbf{y}}_i = (\tilde{y}_{i,1}, \ldots, \tilde{y}_{i,m}) \leftarrow \mathsf{CoPIR.Retrieve}(\mathsf{copir}_{2,i}, \mathsf{st}_i)$.
- For $i \in [\ell]$ and $j \in [t]$ compute $\tilde{z}_{i,j} \leftarrow \mathsf{PIR.Retrieve}(\mathsf{r}_{i,j}, \hat{\mathsf{st}}_{i,j})$.
- For $i \in [\ell]$ set $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,m})$ where

$$z_{i,l} = \begin{cases} \tilde{z}_{i,j} & \text{if } l = J_i[j] \\ \tilde{y}_{i,\ell} & \text{otherwise} \end{cases}.$$

- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$.
- Compute $\tilde{\mathbf{W}} \leftarrow \mathsf{LHE.DecShrink}(\mathsf{sk}, \tilde{\mathsf{ct}})$ and $\mathbf{W} = \tilde{\mathbf{W}} - \mathbf{Z}$.
- Let $\mathbf{w}_1, \ldots, \mathbf{w}_\ell$ be the rows of $\mathbf{W}$. Output $\mathbf{w} = (\mathbf{w}_1\| \ldots \|\mathbf{w}_\ell) \in \{0,1\}^{m\ell}$.

Correctness. We will first show that OT is correct, given that LHE, CoPIR and PIR are correct.

Theorem 4.7.1. *Assume that* LHE, CoPIR *and* PIR *are correct. Then the scheme presented above is correct.*

Proof. First, note that by linear-homomorphic correctness of LHE it holds that

$$\begin{aligned}\tilde{\mathbf{W}} &= \mathsf{LHE.DecShrink}(\mathsf{sk}, \mathsf{LHE.Eval\&Shrink}(\mathsf{pk}, f, \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_1), \dots, \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_\ell)) \\ &= f(\mathbf{S}_1, \dots, \mathbf{S}_\ell) \\ &= \left(\sum_{i=1}^{k} (-\mathbf{S}_i\mathbf{A} + \mathbf{C}_i) \cdot \mathbf{D}_i\right) + \mathbf{Z}\end{aligned}$$

Let $\tilde{\mathbf{w}}_i$ be the i-th row of $\tilde{\mathbf{W}}$. It holds by definition $\mathbf{S}_i$, $\mathbf{C}_i$ and $\mathbf{Z}_i$ that

$$\begin{aligned}\tilde{\mathbf{w}}_i &= (-\mathbf{s}_i\mathbf{A} + \mathbf{c}_i)\mathbf{D}_i + \mathbf{z}_i \\ &= (-\mathbf{s}_i\mathbf{A} + \mathbf{s}_i\mathbf{A}_i + \mathbf{e}_i + \mathbf{b}_i)\mathbf{D}_i + \mathbf{m}_{0,i} + \mathbf{y}_i \\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{y}_i.\end{aligned}$$

where $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,m})$ is part of the output of $\mathsf{CoPIR.Send}$.

Let J_i be the support of $\mathbf{e}_i$ and let $\tilde{y}_i = (\tilde{y}_{i,1}, \dots, \tilde{y}_{i,m}) \leftarrow \mathsf{CoPIR.Retrieve}(\mathsf{copir}_{2,i}, \mathsf{st}_i)$. By the correctness of the Co-PIR scheme CoPIR we have that $\tilde{y}_{i,j} = y_{i,j}$ for all $j \notin J_i$. On the other hand, by the correctness of the PIR scheme PIR it holds that

$$\tilde{z}_{i,j} = y_{i,j} + (m_{1,i,j} - m_{0,i,j})$$

for all $j \in J_i$. Consequently, we have that

$$z_{i,j} = \begin{cases} y_{i,j} + (m_{1,i,j} - m_{0,i,j}) & \text{if } l = J_i[j] \\ y_{i,j} & \text{otherwise} \end{cases}.$$

In other words, the term $(m_{1,i,j} - m_{0,i,j})$ only appears in the coordinates where $\mathbf{e}_i$ is equal to one. Then, it holds that

$$\mathbf{z}_i = \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{y}_i.$$

We conclude that

$$\mathbf{w} = \tilde{\mathbf{w}}_i - \mathbf{z}_i = \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i}.$$

Since $\mathbf{w} = (\mathbf{w}_1 \| \dots \| \mathbf{w}_\ell)$ it follows that

$$\mathbf{w} = \mathbf{b} \odot (\mathbf{m}_1 - \mathbf{m}_0) + \mathbf{m}_0,$$

i.e. OT is correct. □

Communication complexity. We will now analyze the communication complexity of OT and show which choice of parameters leads to an overall rate approaching 1.

The bit-size of the message $\mathsf{ot}_1 = \left(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}_{1,i}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\right)$ can be bounded as follows.

- $|\mathsf{pk}| = \ell \cdot \mathsf{poly}(\lambda)$
- $|\mathbf{A}| = n \cdot m$
- $|\{\mathsf{ct}_i\}_{i\in[\ell]}| = \ell^2 \cdot n \cdot \mathsf{poly}(\lambda)$
- $|\{\mathbf{c}_i\}_{i\in[\ell]}| = \ell \cdot m$
- $|\{\mathsf{copir}_{1,i}\}_{i\in[\ell]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$.

Consequently, the overall upload-rate ρ_{up} can be bounded by

$$\begin{aligned}\rho_{\text{up}} &= \frac{|\mathsf{pk}| + |\mathbf{A}| + |(\mathsf{ct}_i)_{i\in[\ell]}| + |(\mathbf{c}_i)_{i\in[\ell]}| + |\{\mathsf{copir}_{1,i}\}_{i\in[\ell]}| + |(\mathsf{q}_{i,j})_{i\in[\ell],j\in[t]}|}{\ell m} \\ &\leq 1 + \frac{\mathsf{poly}(\lambda)}{m} + \frac{n}{\ell} + \frac{\ell\cdot n\cdot \mathsf{poly}(\lambda)}{m} + \frac{t\cdot \mathsf{polylog}(m)\cdot \mathsf{poly}(\lambda)}{m} \\ &\leq 1 + \frac{n}{\ell} + \frac{\ell\cdot n\cdot \mathsf{poly}(\lambda)}{m} + \frac{t\cdot \mathsf{polylog}(m)\cdot \mathsf{poly}(\lambda)}{m}.\end{aligned}$$

We get an overall upload rate of $\rho_{\text{up}} = 1 + O(1/\lambda)$ by choosing $\ell = \lambda \cdot n$ and $m = n^2 \cdot \mathsf{poly}(\lambda)$ for a sufficiently large $\mathsf{poly}(\lambda)$ depending on ε (where $t = m^{1-\varepsilon}$).

The bit-size of the message $\mathsf{ot}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\right)$ can be bounded as follows.

- $|\tilde{\mathsf{ct}}| = \ell m(1 + \rho_{\mathsf{LHE}})$, where $1 + \rho_{\mathsf{LHE}}$ is the ciphertext rate of LHE.
- $|\{\mathsf{copir}_{2,i}\}_{i\in[\ell]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$

Thus, the download-rate ρ_{down} can be bounded by

$$\rho_{\text{down}} = \frac{|\tilde{\mathsf{ct}}| + |\{\mathsf{copir}_{2,i}\}_{i\in[\ell]}| + |\{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}|}{\ell m} \leq 1 + \rho_{\mathsf{LHE}} + \frac{\ell\cdot t\cdot \mathsf{polylog}(m)\cdot \mathsf{poly}(\lambda)}{m}.$$

By the above choice of m this comes down to $\rho_{\text{down}} \leq 1 + \rho_{\mathsf{LHE}} + O(1/\lambda)$.

4.7.2 Security

Receiver Security We now focus on the security of the scheme. We start by proving that the scheme is secure against semi-honest senders.

Theorem 4.7.2. *Assume that* LHE *is a semantic secure LHE scheme,* PIR *is a user-private PIR scheme,* CoPIR *is a receiver secure Co-PIR scheme and that the* $\mathsf{LPN}(n, m, \rho)$ *assumption holds for* $\rho = m^{1-\varepsilon}$ *for* $\varepsilon > 0$*. Then the scheme presented in Section 4.7.1 is receiver secure against semi-honest adversaries.*

Recall that the receiver's message is composed of LHE ciphertexts, LPN samples, Co-PIR and PIR first messages. In a nutshell, receiver security follows from the fact that the ciphertexts hide the LPN secret, the LPN samples hide the receiver's input $\mathbf{b}$ and finally the Co-PIR and PIR first messages hide the indices J_i.

Proof. We first present the simulator for the semi-honest sender. The simulator Sim receives the sender's input and sends it to the ideal functionality. Then it simulates the receiver as follows:

$\mathsf{Sim}(1^\lambda)$:

- Choose $\mathbf{A} \leftarrow\$ \{0,1\}^{n\times m}$ and compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, \ell)$.
- For all $i \in [\ell]$, choose $\mathbf{c}_i \leftarrow\$ \{0,1\}^m$ and compute $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, 0)$.
- For all $i \in [\ell]$, let $J_i \subset [m]$ be a random subset of size t. Compute $\mathsf{copir}_{1,i} \leftarrow \mathsf{CoPIR}(J_i)$. Additionally, for $j \in [t]$ compute $(\mathsf{q}_{i,j}, \hat{\mathsf{st}}_{i,j}) = \mathsf{PIR.Query}(a_{i,j})$ where $a_{i,j} \leftarrow\$ [m]$.
- Output $\mathsf{ot}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell], j\in[t]}\big)$.

We now show that the ideal world and real-world executions are indistinguishable. The proof follows from the following sequence of hybrids.

Hybrid $\mathcal{H}_0$**.** This hybrid is the real experiment.

For $i \in [\ell]$, consider the following sub-hybrids.

Hybrid $\mathcal{H}_{1,i}$**.** Let $\mathcal{H}_{1,0} = \mathcal{H}_0$. This hybrid is identical to the previous one, except that the receiver computes $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, 0)$.

Indistinguishability of hybrids $\mathcal{H}_{1,i-1}$ and $\mathcal{H}_{1,i}$ are indistinguishable, for $i = 1, \dots, \ell$ and where $\mathcal{H}_{1,0} = \mathcal{H}_0$. follows from the semantic security of LHE.

Hybrid $\mathcal{H}_{2,i}$**.** Let $\mathcal{H}_{2,0} = \mathcal{H}_{1,\ell}$. This hybrid is identical to the previous one, except that the receiver computes $(\mathsf{copir}_{1,i}, \mathsf{st}_i) = \mathsf{CoPIR.Query}(J'_i[j])$ where J'_i is a uniform subset of $[m]$ of size t.

Indistinguishability of hybrids $\mathcal{H}_{2,i-1}$ and $\mathcal{H}_{2,i}$, for $i = 1, \dots, \ell$ and where $\mathcal{H}_{2,0} = \mathcal{H}_{1,\ell}$, follows directly from the receiver security of the underlying CoPIR.

Let $\varphi : [\ell t] \to [\ell] \times [t]$ be a bijective function. For $i' \in [\ell t]$ consider the following hybrids.

Hybrid $\mathcal{H}_{3,i'}$. Let $\mathcal{H}_{3,0} = \mathcal{H}_{2,\ell}$. Let $\varphi(i') = (i,j)$. This hybrid is identical to the previous one, except that the receiver computes $(\mathsf{q}_{i,j}, \hat{\mathsf{st}}_{i,j}) = \mathsf{PIR.Query}(a_{i,j})$ where $a_{i,j} \leftarrow\$ [m]$.

Indistinguishability of hybrids $\mathcal{H}_{3,i-1}$ and $\mathcal{H}_{3,i}$, for $i = 1, \dots, \ell t$ and where $\mathcal{H}_{3,0} = \mathcal{H}_{2,\ell}$, follows directly from the receiver security of the underlying PIR.

Finally for $i \in [\ell]$ consider the following sub-hybrids.

Hybrid $\mathcal{H}_{4,i}$. Let $\mathcal{H}_{4,0} = \mathcal{H}_{3,\ell t}$. This hybrid is identical to the previous one, except that the receiver samples $\mathbf{c}_i \leftarrow\$ \{0,1\}^m$.

Indistinguishability of hybrids $\mathcal{H}_{4,i-1}$ and $\mathcal{H}_{4,i}$, for $i = 1, \dots, \ell$ and where $\mathcal{H}_{4,0} = \mathcal{H}_{3,\ell t}$, follows directly from the LPN assumption.

Finally, note that hybrid $\mathcal{H}_{4,\ell}$ is identical to the ideal-world execution. This concludes the proof of receiver security. □

Sender Security

Theorem 4.7.3. *Assume that* LHE *is a statistically function-private LHE scheme,* PIR *is a sender-private PIR scheme and* CoPIR *is a sender-private Co-PIR scheme. Then the scheme presented in Section 4.7.1 is sender secure.*

Proof. We begin by presenting the simulator Sim against a semi-honest receiver. Recall that, in the semi-honest case, the simulator has access to the receiver's internal state. The simulator sends $\mathbf{b} = (b_1, \dots, b_{m\ell})$ to the ideal functionality and receives $\tilde{\mathbf{m}} = (\tilde{m}_1, \dots, \tilde{m}_{m\ell})$.

$\mathsf{Sim}(1^\lambda, \tilde{\mathbf{m}} \in \{0,1\}^{m\ell}, \mathbf{e} \in \{0,1\}^{m\ell}, \mathsf{ot}_1)$:

- Parse $\mathsf{ot}_1 = \left(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell], j\in[t]}\right)$.
- It sets $m_{b_i,i} = \tilde{m}_i$ and $m_{1-b_i,i} = 0$. Finally, it sets $\mathbf{m}_0 = (m_{0,1}, \dots, m_{0,m\ell})$ and $\mathbf{m}_1 = (m_{1,1}, \dots, m_{1,m\ell})$.
- For $i \in [\ell]$, let $J_i = \mathsf{Supp}(\mathbf{e}_i) := \{J_i[1], \dots, J_i[t]\}$. Compute $(\mathsf{copir}_{2,i}, \mathbf{y}_i) \leftarrow \mathsf{CoPIR}(\mathsf{copir}_{1,i})$ where $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,m})$.
- For $i \in [\ell]$ and $j \in [t]$, choose $y'_{i,J_i[j]} \leftarrow\$ \{0,1\}$ restricted to $y_{i,J_i[j]} = y'_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]})$. Set $\overline{\mathbf{DB}}_{i,j} = (0, \dots, y'_{i,J_i[j]}, \dots, 0)$. Compute $\mathbf{r}_{i,j} \leftarrow \mathsf{PIR.Send}(DB_{i,j}, \mathsf{q}_{i,j})$.
- Compute $\tilde{\mathsf{ct}} \leftarrow \mathsf{LHE.Sim}(\mathsf{pk}, \mathbf{w}^*)$ where $\mathbf{w}^* := (\mathbf{w}^*_1, \dots, \mathbf{w}^*_m)$ and $\mathbf{w}^*_i = \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{z}'_i$. Here, $\mathbf{z}'_i = (z_{i,1}, \dots, z_{i,m})$ such that

$$z'_{i,j'} = \begin{cases} y'_{i,j} & \text{if } j' = J_i[j] \\ y_{i,j'} & \text{otherwise} \end{cases}$$

- Output $\mathsf{ot}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\right)$.

We will establish security via the following sequence of hybrids to show that the ideal-world experiment and the real-world one are indistinguishable.

Hybrid $\mathcal{H}_0$. This is the real experiment.

For $i' \in [\ell t]$ consider the following sub-hybrid. Let $\varphi : [\ell t] \to [\ell] \times [t]$.

Hybrid $\mathcal{H}_{1,i'}$. Let $\mathcal{H}_{1,0} = \mathcal{H}_0$. Let $\varphi(i') = (i,j)$. This hybrid is identical to the previous one except that we set $\overline{\mathbf{DB}}_{i,j} = (0, \ldots, y_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]}), \ldots, 0)$, i.e, $\overline{\mathbf{DB}}_{i,j}$ is set to 0 everywhere except for position $J_i[j]$ where it assumes the value $y_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]})$ as in the previous hybrid. Additionally, we compute $\mathsf{r}_{i,j} \leftarrow \mathsf{PIR.Send}(\overline{\mathbf{DB}}_{i,j}, \mathsf{q}_{i,j})$.

Indistinguishability of hybrids $\mathcal{H}_{1,i'-1}$ and $\mathcal{H}_{1,i'}$ follows from the sender security of PIR.

For $i \in [\ell]$ consider the following hybrid.

Hybrid $\mathcal{H}_{2,i}$. Let $\mathcal{H}_{2,0} = \mathcal{H}_{1,\ell t}$. This hybrid is identical to the previous one except that for all $j \in [t]$, choose $y'_{i,J_i[j]} \leftarrow \{0,1\}$ such that $y_{i,J_i[j]} = y'_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]})$.

Indistinguishability of hybrids $\mathcal{H}_{2,i-1}$ and $\mathcal{H}_{2,i}$ follows from the sender security of CoPIR.

Note that, in this hybrid $\overline{\mathbf{DB}}_{i,j}$ is of the form

$$\overline{\mathbf{DB}}_{i,j} = (0, \ldots, y'_{i,J_i[j]}, \ldots, 0).$$

Furthermore, note that we can write $\mathbf{z}_i$ as $\mathbf{z}_i = \mathbf{z}'_i - \mathbf{e} \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i})$, where $\mathbf{z}'_i = (z'_{i,1}, \ldots, z'_{i,m})$ is defined by

$$z'_{i,j'} = \begin{cases} y'_{i,j} & \text{if } j' = J_i[j] \\ \tilde{y}_{i,j'} & \text{otherwise} \end{cases}$$

where $\tilde{y}_{i,j} = y_{i,j}$ for $j \notin J_i$ by the correctness of CoPIR.

Finally, consider the remaining sub-hybrids.

Hybrid $\mathcal{H}_3$. In this hybrid we compute ct as follows: Set $\mathbf{W}^* \leftarrow f(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and compute $\mathsf{ct} \leftarrow \mathsf{LHE.Sim}(\mathsf{pk}, \mathbf{W}^*)$, where $\mathsf{LHE.Sim}$ is the function-privacy simulator for LHE. Statistical indistinguishability between $\mathcal{H}_{2,\ell}$ and $\mathcal{H}_3$ follows from the statistical function privacy of LHE.

Hybrid $\mathcal{H}_4$. In this hybrid, we compute $\mathbf{W}^* = (\mathbf{w}^*_1, \ldots, \mathbf{w}^*_m)$ via

$$\mathbf{w}^*_i = \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{z}'_i.$$

Finally, note that

$$\begin{aligned}\mathbf{W}^* &= f(\mathbf{S}_1, \dots, \mathbf{S}_\ell)\\ &= \left(\sum_{i=1}^{k}(-\mathbf{S}_i\mathbf{A} + \mathbf{C}_i) \cdot \mathbf{D}_i\right) + \mathbf{Z}.\end{aligned}$$

Let $\mathbf{w}_i^*$ be the i-th row of $\mathbf{W}^*$. It holds by definition $\mathbf{S}_i$, $\mathbf{C}_i$ and $\mathbf{Z}_i$ that

$$\begin{aligned}\mathbf{w}_i^* &= (-\mathbf{s}_i\mathbf{A} + \mathbf{c}_i)\mathbf{D}_i + \mathbf{z}_i\\ &= (-\mathbf{s}_i\mathbf{A} + \mathbf{s}_i\mathbf{A}_i + \mathbf{e}_i + \mathbf{b}_i)\mathbf{D}_i + \mathbf{m}_{0,i} + \mathbf{z}_i\\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{z}_i\\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + (\mathbf{z}_i' - \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}))\\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{z}_i'\end{aligned}$$

Consequently, $\mathcal{H}_4$ is identical to the ideal experiment. □

Hardness assumptions for optimal-rate OT. When we instantiate the LHE with one of the schemes from Section 4.5, the Co-PIR with the construction from Section 4.6 and the PIR with a known black-box construction based on LWE, DDH or QR [DGI+19], we obtain the following corollary

Corollary 4.7.4. *Assuming the LWE, DDH or QR assumptions together with the* $\mathsf{LPN}(n, m, \rho)$*, there is a black-box construction for optimal-rate OT.*

4.8 Oblivious Matrix-Vector Product and Oblivious Linear Evaluation

In this section, we show how we can extend the techniques from the previous section to build protocols for OMV and OLE that achieve optimal rates.

4.8.1 OMV Protocol

We start by presenting a secure protocol for oblivious matrix-vector products (OMV). In an OMV functionality, there is a sender, with input a matrix $\mathbf{M} \in \mathbb{Z}_q^{m\times m}$ and a vector $\mathbf{v} \in \mathbb{Z}_q^m$, and a receiver with input $\mathbf{b} \in \mathbb{Z}_q^m$. In the end, the receiver gets the value $\mathbf{bM} + \mathbf{v}$ but learns nothing about $\mathbf{M}$ and $\mathbf{v}$ whereas the sender learns nothing about $\mathbf{b}$.

We start by defining the functionality:

OMV functionality. The functionality $\mathcal{F}_{\mathsf{OMV}}$ is parametrized by integers $m = \mathsf{poly}(\lambda)$ and q and works as follows:

- **Receiver phase.** R sends $\mathbf{b}$ to $\mathcal{F}_{\mathsf{OMV}}$ where $\mathbf{b} \in \mathbb{Z}_q^m$.
- **Sender phase.** S sends $(\mathbf{M}, \mathbf{v})$ to $\mathcal{F}_{\mathsf{OMV}}$ where $\mathbf{M} \in \mathbb{Z}_q^{m \times m}$ and $\mathbf{v} \in \{0,1\}^m$. $\mathcal{F}_{\mathsf{OMV}}$ sends $\mathbf{bM} + \mathbf{v} \in \mathbb{Z}_q^m$ to R.

Below, we present a protocol for OMV that supports a sublinear number of multiplications in the size of the matrix. That is, all columns and rows of the matrix M should have bounded (sublinear in m) hamming weight.[12]

The Protocol

We start by presenting the ingredients that we need for our OMV protocol.

Ingredients. Let $q = \mathsf{poly}(\lambda)$. We will need the following ingredients.

- A packed linearly homomorphic encryption scheme $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Shrink}, \mathsf{DecShrink})$ with plaintext space $\mathbb{Z}_q^\ell$ and a post-homomorphism shrinking procedure Shrink which converts ciphertexts into a rate 1 representation.
- The binary $\mathsf{LPN}(n, m, \rho, q)$ problem with dimension $n = \mathsf{poly}(\lambda)$, $m = n \cdot \ell \cdot \mathsf{poly}(\lambda)$ samples and slightly sub-constant noise-rate $\rho = m^{1-\varepsilon}$.
- A 2-round PIR scheme $\mathsf{PIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ with poly-logarithmic communication complexity and sender privacy.
- A 2-round Co-PIR scheme $\mathsf{CoPIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ over $\mathbb{Z}_q$ parametrized by $m(q-1)$.

We define the hamming weight of a matrix $\mathbf{D} \in \mathbb{Z}_q^{m \times m}$ to be the value $\mathsf{hw}(\mathbf{D}) = \max_i\{\mathsf{hw}(\mathbf{d}_i)\}, \mathsf{hw}(\mathbf{d}^{(i)})\}$ for all $i \in [m]$, where $\mathbf{d}_i$ and $\mathbf{d}^{(i)}$ are the i-th row and column of $\mathbf{D}$ respectively. In addition to the notation presented in Section 4.7, we present the following algorithm:

$\mathsf{AffineDecomp}(\mathbf{D} \in \mathbb{Z}_q^{m \times m})$: Takes a matrix $\mathbf{D}$ such that $\mathsf{hw}(\mathbf{D}) \leq \mu$ for all $i \in [m]$. It outputs $\mathbf{T}_1, \ldots, \mathbf{T}_\mu \in \mathbb{Z}_q^{m \times m}$ such that $\mathsf{hw}(\mathbf{T}_i) \leq 1$ for all $i \in [\mu]$ and $\mathbf{D} = \mathbf{T}_1 + \cdots + \mathbf{T}_\mu$.

Protocol. The protocol $\mathsf{OMV} = (\mathsf{OMVR}, \mathsf{OMVS}, \mathsf{OMVD})$ is presented below.

$\mathsf{OMVR}(\mathbf{b} \in \mathbb{Z}_q^{m\ell})$:

- Parse $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$, where the $\mathbf{b}_i \in \mathbb{Z}_q^m$ are blocks of size m.
- Choose $\mathbf{A} \leftarrow\!\$\ \mathbb{Z}_q^{n \times m}$ uniformly at random and compute a pair of public and secret key $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LHE}.\mathsf{KeyGen}(1^\lambda, \ell)$.

[12] Recall that hamming weight is used to count non-zero elements.

- For all $i \in [\ell]$, choose $\mathbf{s}_i \leftarrow\$ \mathbb{Z}_q^n$, and $\mathbf{e}_i \leftarrow\$ \chi_{m,t,q}$, compute $\mathbf{c}_i \leftarrow \mathbf{s}_i\mathbf{A} + \mathbf{e}_i + \mathbf{b}_i$, and set $\mathbf{S}_i \leftarrow \mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{s}_i)$. Compute a matrix-ciphertext $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_i)$.
- For all $i \in [\ell]$ set $J_i = \mathsf{Supp}(\mathbf{e}_i)$ to be the support of $\mathbf{e}_i$.
- For all $i \in [\ell]$ and $k \in [\mu]$ compute $(\mathsf{copir}_{1,i,k}, \mathsf{st}_{i,k}) \leftarrow \mathsf{CoPIR.Query}(J_i)$. Additionally, for all $j \in [t]$ compute $(\mathsf{q}_{i,k,j}, \hat{\mathsf{st}}_{i,k,j}) = \mathsf{PIR.Query}((q-1)(J_i[j]-1) + e_{i,J_i[j]})$.
- Output $\mathsf{omv}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i\}_{i\in[\ell]}, \{\mathsf{copir}_{1,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{q}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\big)$ and

$$\mathsf{st} = \left(\mathsf{sk}, \{J_i\}_{i\in[\ell]}, \{\mathsf{st}_{i,k}\}_{i\in[\ell],k\in[\mu]}, \{\hat{\mathsf{st}}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\right).$$

$\mathsf{OMVS}((\mathbf{D}, \mathbf{v}) \in \mathbb{Z}_q^{m\times m\ell} \times \mathbb{Z}_q^{m\ell}, \mathsf{omv}_1)$:

- Parse $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_\ell)$ and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_\ell)$. If $\mathsf{hw}(\mathbf{D}) > \mu$ abort the protocol. Parse $\mathsf{omv}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i\}_{i\in[\ell]}, \{\mathsf{copir}_{1,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{q}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\big)$.
- For $i \in [\ell]$ and $k \in [\mu]$ $(\mathbf{y}_{i,k}, \mathsf{copir}_{2,i,k}) \leftarrow \mathsf{CoPIR.Send}(\mathsf{copir}_{1,i,k})$ where $\mathbf{y}_{i,k} = (y_{i,k,1}, \ldots, y_{i,k,m})$.
- For all $i \in [\ell]$ set $\mathbf{z}_i = \mathbf{v}_i + \sum_{k=1}^{\mu} \mathbf{y}_{i,k}$.
- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$.
- For all $i \in [\ell]$ set $\mathbf{C}_i = \mathsf{SingleRowMatrix}(\ell, m, i, \mathbf{c}_i)$.
- Define the $\mathbb{Z}_q$-linear function $f \colon (\mathbb{Z}_q^{\ell\times n})^\ell \to \mathbb{Z}_q^{\ell\times m}$ via

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_\ell) = \left(\sum_{i=1}^{\ell} (-\mathbf{X}_i\mathbf{A} + \mathbf{C}_i) \cdot \mathbf{D}_i\right) + \mathbf{Z}.$$

- Compute $\tilde{\mathsf{ct}} \leftarrow \mathsf{LHE.Eval\&Shrink}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
- For all $\in [\ell]$, set $(\mathbf{T}_{i,1}, \ldots, \mathbf{T}_{i,\mu}) \leftarrow \mathsf{AffineDecomp}(\mathbf{D}_i)$. Moreover, for all $k \in [\mu]$ and all $l \in [m]$, let $t_{i,k,l}$ be the only non-zero element in the l-th row of $\mathbf{T}_{i,k}$. If its l-th row is a zero vector, set $t_{i,k,l} = 0$.
- For all $i \in [\ell]$ and $k \in [\mu]$ set

  $DB_{i,k} = (y_{i,k,1}+t_{i,k,1}, y_{i,k,1}+2\cdot t_{i,k,1}, \ldots, y_{i,k,1}+(q-1)\cdot t_{i,k,1}, \ldots, y_{i,k,m}+(q-1)\cdot t_{i,k,m})$,

  where $DB_{i,k}$ is a $(q-1)m$-sized vector. For all $j \in [t]$ compute $\mathsf{r}_{i,k,j} \leftarrow \mathsf{PIR.Send}(DB_{i,k}, \mathsf{q}_{i,k,j})$.
- Output $\mathsf{omv}_2 = \big(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{r}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\big)$.

$\mathsf{OMVD}(\mathsf{omv}_2, \mathsf{st})$:

- Parse $\mathsf{omv}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{r}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\right)$ and
$$\mathsf{st} = \left(\mathsf{sk}, \{J_i\}_{i\in[\ell]}, \{\mathsf{st}_{i,k}\}_{i\in[\ell],k\in[\mu]}, \{\hat{\mathsf{st}}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\right).$$
- For all $i \in [\ell]$ and $k \in [\mu]$ compute $\tilde{\mathbf{y}}_{i,k} = (\tilde{y}_{i,k,1}, \dots, \tilde{y}_{i,k,m}) \leftarrow \mathsf{CoPIR.Retrieve}(\mathsf{copir}_{2,i,k}, \mathsf{st}_{i,k})$.
- For $i \in [\ell], k \in [\mu]$ and $j \in [t]$ compute $\tilde{z}_{i,k,j} \leftarrow \mathsf{PIR.Retrieve}(\mathsf{r}_{i,k,j}, \hat{\mathsf{st}}_{i,k,j})$.
- For all $i \in [\ell]$ and $k \in [\mu]$ set $\mathbf{z}_{i,k} = (z_{i,k,1}, \dots, z_{i,k,m})$ where
$$z_{i,k,l} = \begin{cases} \tilde{z}_{i,k,j} & \text{if } l = J_i[j] \\ \tilde{y}_{i,k,l} & \text{otherwise} \end{cases}.$$
- For all $i \in [\ell]$ set $\mathbf{z}_i = \sum_{k=1}^{\mu} \mathbf{z}_{i,k}$.
- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \dots, \mathbf{z}_\ell)$.
- Compute $\tilde{\mathbf{W}} \leftarrow \mathsf{LHE.DecShrink}(\mathsf{sk}, \tilde{\mathsf{ct}})$ and $\mathbf{W} = \tilde{\mathbf{W}} - \mathbf{Z}$.
- Let $\mathbf{w}_1, \dots, \mathbf{w}_\ell$ be the rows of $\mathbf{W}$. Output $\mathbf{w} = (\mathbf{w}_1 \| \dots \| \mathbf{w}_\ell) \in \mathbb{Z}_q^{m\ell}$.

Correctness. We first show that the scheme presented above is correct.

Theorem 4.8.1 (Correctness). *Assume that* LHE*,* CoPIR *and* PIR *are correct. Then the scheme presented above is correct.*

The proof follows the same reasoning as the proof of Theorem 4.7.1.

Communication complexity. We now analyze the communication complexity of OMV and show which choice of parameters leads to an overall rate approaching 1.

The bit-size of the message $\mathsf{omv}_1 = \left(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i\}_{i\in[\ell]}, \{\mathsf{copir}_{1,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{q}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\right)$ can be bounded as follows:

- $q = \mathsf{poly}(\lambda)$
- $|\mathsf{pk}| = \ell \cdot \mathsf{poly}(\lambda)$
- $|\mathbf{A}| = n \cdot m \cdot \log q$
- $|\{\mathsf{ct}_i\}_{i\in[\ell]}| = \ell^2 \cdot n \cdot \mathsf{poly}(\lambda)$
- $|\{\mathbf{c}_i\}_{i\in[\ell]}| = \ell \cdot m \cdot \log q$
- $|\{\mathsf{copir}_{1,i,k}\}_{i\in[\ell]}| = \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m, q) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{q}_{i,k,j}\}_{i\in[\ell],j\in[t]}| = q \cdot \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$.

Thus, the overall upload-rate ρ_{up} can be bounded by

$$\rho_{\text{up}} \leq 1 + \frac{n \log q}{\ell} + \frac{\ell \cdot n \cdot \mathsf{poly}(\lambda)}{m} + \frac{\mu \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)}{m}.$$

We get an overall upload rate of $\rho_{\text{up}} = 1 + O(1/\lambda)$ by choosing $\ell = \lambda \cdot n \log q$, $\mu = m^{1-\zeta}$ (for some $\zeta > 0$ such that $\zeta + \varepsilon > 1$) and $m = n^2 \cdot \log q \cdot \mathsf{poly}(\lambda)$ for a sufficiently large $\mathsf{poly}(\lambda)$ depending on ε (where $t = m^{1-\varepsilon}$).

The bit-size of the message $\mathsf{omv}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i,k}\}_{i \in [\ell], k \in [\mu]}, \{\mathsf{r}_{i,k,j}\}_{i \in [\ell], k \in [\mu], j \in [t]}\right)$ can be bounded as follows:

- $q = \mathsf{poly}(\lambda)$
- $|\tilde{\mathsf{ct}}| = \ell m(1 + \rho_{\mathsf{LHE}})$, where $1 + \rho_{\mathsf{LHE}}$ is the ciphertext rate of LHE
- $|\{\mathsf{copir}_{2,i,k}\}_{i \in [\ell]}| = \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{r}_{i,k,j}\}_{i \in [\ell], j \in [t]}| = q \cdot \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$.

Thus, the download-rate ρ_{down} can be bounded by

$$\rho_{\text{down}} \leq 1 + \rho_{\mathsf{LHE}} + \frac{\mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)}{m}.$$

By the above choice of m and μ this comes down to $\rho_{\text{down}} \leq 1 + \rho_{\mathsf{LHE}} + O(1/\lambda)$.

Security. Finally, we state the result that guarantees security of the scheme.

Theorem 4.8.2 (Security). *The scheme presented above is:*

- *Receiver secure if* LHE *is a semantic secure LHE scheme,* PIR *is a user-private PIR scheme,* CoPIR *is a receiver secure Co-PIR scheme and that the* $\mathsf{LPN}(n, m, \rho, q)$ *assumption holds for* $\rho = m^{1-\varepsilon}$ *for* $\varepsilon > 0$*.*
- *Sender secure if* LHE *is a statistically function-private LHE scheme,* PIR *is a sender-private PIR scheme and* CoPIR *is a sender-private Co-PIR scheme.*

The proof of the theorem follows the same reasoning as the proof of Theorem 4.7.2 and Theorem 4.7.3.

Again, instantiating the ingredients used in OMV with the constructions from this chapter, we obtain the following corollary.

Corollary 4.8.3. *There exists a black-box construction for* OMV *over* $\mathbb{Z}_q$ *assuming:*

- *LWE and* $\mathsf{LPN}(n, m, \rho, q)$ *for* $q = \mathsf{poly}(\lambda)$
- *DDH and* $\mathsf{LPN}(n, m, \rho, q)$ *for* $q = 2^{\mu} = \mathsf{poly}(\lambda)$*.*

4.8.2 OLE Protocol

An oblivious linear evaluation (OLE) is a protocol between a sender, with input an affine function f, and a receiver, with input a point b. It allows for the receiver to obliviously learn $f(b)$. We now show how we can obtain an OLE using the OMV protocol presented in Section 4.8.1.

We start by defining the functionality:

OLE functionality. The functionality $\mathcal{F}_{\mathsf{OLE}}$ is parametrized by integers $k = \mathsf{poly}(\lambda)$ and q and works as follows:

- **Receiver phase.** R sends $\mathbf{b}$ to $\mathcal{F}_{\mathsf{OLE}}$ where $\mathbf{b} \in \mathbb{Z}_q^k$.
- **Sender phase.** S sends $(\mathbf{u}_0, \mathbf{u}_1)$ to $\mathcal{F}_{\mathsf{OLE}}$ where $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{Z}_q^k$. $\mathcal{F}_{\mathsf{OLE}}$ sends $\mathbf{b} \odot \mathbf{u}_0 + \mathbf{u}_1 \in \mathbb{Z}_q^k$ to R.

Protocol for Small Fields

We briefly sketch how we can construct an OLE scheme over $\mathbb{Z}_q$ where $q = \mathsf{poly}(\lambda)$. The protocol follows as a particular case of the protocol of Section 4.8.1. We give a brief overview of the scheme below.

Let Using the notation of Section 4.8.1, let $\mathbf{b} = (\mathbf{b}_1, \dots, \mathbf{b}_\ell) \in \mathbb{Z}_q^{m\ell}$ be the receiver's input and let $(\mathbf{u}_0 = (\mathbf{u}_{0,1}, \dots, \mathbf{u}_{0,\ell}), \mathbf{u}_1 = (\mathbf{u}_{1,1}, \dots, \mathbf{u}_{1,\ell})) \in (\mathbb{Z}_q^{m\ell})^2$ be the sender's input. To achieve OLE, the sender constructs the matrices $\mathbf{D}_i = \mathsf{Diag}(m, \mathbf{u}_{0,i})$ and sets $\mathbf{v}_i = \mathbf{u}_{1,i}$ for all $i \in [\ell]$. Then they run the OMV protocol where the receiver inputs $\mathbf{b}$ and the sender inputs $\mathbf{D} = (\mathbf{D}_1, \dots, \mathbf{D}_\ell)$ and $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_\ell)$. It is easy to see that the output of the receiver is $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_\ell)$ where

$$\mathbf{y}_i = \mathbf{b}_i \mathbf{D}_i + \mathbf{v}_i = \mathbf{b}_i \odot \mathbf{u}_{0,i} + \mathbf{u}_{1,i}$$

be the correctness of the OMV protocol.

Moreover, $\mathsf{hw}(\mathbf{D}_i) = 1 \leq m^{1-\zeta}$ for some $\zeta > 0$ such that $\zeta + \varepsilon > 1$. Thus the resulting protocol achieves overall rate 1. Finally, in terms of hardness assumptions, the OLE protocol inherits the same security.

Extending OLE to Larger Rings

Following [DGI⁺19], we briefly explain how we can achieve OLE over larger rings (which can potentially have a super-polynomial size in λ).

OLE over $\mathbb{Z}_N = \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_\delta}$. Let $N = \prod_{i=1}^{\delta} q_i$ be an integer (which might be superpolynomial in λ) such that for all $i \in [\delta]$ $q_i = \mathsf{poly}(\lambda)$ are different prime numbers. Then, via the Chinese Remainder Theorem, $\mathbb{Z}_N$ is isomorphic to $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_\delta}$. Thus, performing an OLE over $\mathbb{Z}_N$ boils down to performing δ OLEs over each one of the smaller fields $\mathbb{Z}_{q_i}$. It is easy to see that, if each OLE over $\mathbb{Z}_{q_i}$ has an overall rate-1, then the resulting OLE over $\mathbb{Z}_N$ also achieves an overall rate-1.

OLE OVER EXTENSION FIELDS. We now show how these techniques can be adapted to perform OLE over an extension field $\mathbb{F}_{q^k}$ of order q^k for a prime q. Here, we rely on the fact that multiplication over $\mathbb{F}_{q^k}$ can be expressed as a linear function over the field $\mathbb{Z}_q$. That is, suppose that an element $\mathbf{x} \in \mathbb{F}_{q^k}$ is of the form $\mathbf{x} = x_1 + x_2\alpha + \cdots + x_k\alpha^{k-1}$ where each $x_i \in \mathbb{Z}_q$ and α is a symbol. Then, for elements $\mathbf{a}, \mathbf{x} \in \mathbb{F}_{q^k}$ the product

$$\mathbf{xa} = f_{1,\mathbf{a}}(\mathbf{x}) + f_{2,\mathbf{a}}(\mathbf{x})\alpha + \cdots + f_{k,\mathbf{a}}(\mathbf{x})\alpha^{k-1}$$

where each $f_{i,\mathbf{a}}$ is a $\mathbb{Z}_q$-linear function which depends solely on $\mathbf{a}$.

Given this, we briefly describe how we can perform several OLEs over $\mathbb{F}_{q^k}$ while preserving overall rate 1. The receiver has input $\mathbf{b} = (\mathbf{b}_1, \dots, \mathbf{b}_t) \in \mathbb{F}_{q^k}^t$ such that $kt = m\ell$ and $k|m$ (using the same notation as in Section 4.8.1). It parses each $\mathbf{b}_i$ as a k-dimensional vectors $\bar{\mathbf{b}}_i \in \mathbb{Z}_q^k$. Then, it organizes all t vectors $\mathbf{b}_i$ in blocks $\mathbf{c}_i \in \mathbb{Z}_q^m$ of size m. It inputs $\mathbf{c} = (\mathbf{c}_1, \dots, \mathbf{c}_\ell)$ into the OMV protocol.

The sender, with input $\mathbf{u}, \mathbf{v} \in \mathbb{F}_{q^k}$ rearranges $\mathbf{u}, \mathbf{v}$ in the same way as the receiver and obtains $\mathbf{w} = (\mathbf{w}_1, \dots, \mathbf{w}_\ell), \mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_\ell)$ respectively. Then, for each $\mathbf{w}_i = (\mathbf{w}_{i,1}, \dots, \mathbf{w}_{i,m/k})$, it computes the functions $f_{j,\mathbf{w}_{i,r}}$ for each $j \in [k]$, $i \in [\ell]$ and $r \in [m/k]$. Let $\mathbf{f}_{j,\mathbf{w}_{i,r}}$ be the vector composed by the coefficients of $f_{j,\mathbf{w}_{i,r}}$. The sender computes the matrices

$$\bar{\mathbf{D}}_{i,r} = \begin{pmatrix} | & & | \\ \mathbf{f}_{1,\mathbf{w}_{i,r}} & \cdots & \mathbf{f}_{k,\mathbf{w}_{i,r}} \\ | & & | \end{pmatrix}$$

and then sets

$$\mathbf{D}_i = \begin{pmatrix} \bar{\mathbf{D}}_{i,1} & & \\ & \ddots & \\ & & \bar{\mathbf{D}}_{i,m/k} \end{pmatrix}.$$

It inputs $\mathbf{D} = (\mathbf{D}_1, \dots, \mathbf{D}_\ell)$ and $\mathbf{z}$ into the OMV protocol.

It is easy to see that the receiver's output will be $\mathbf{b} \odot \mathbf{u} + \mathbf{v}$ where $\odot$ denotes component-wise multiplication over $\mathbb{F}_{q^k}$. Moreover, $\mathsf{hw}(\mathbf{D}_i) = k$. By choosing k such that $k \leq \mu = m^{1-\zeta}$ we achieve a protocol with overall rate 1. In particular, we can set the parameters such that $k = \lambda$ and we achieve an OLE over the field $\mathbb{F}_{q^\lambda}$ of exponential size.

5 Privacy Preserving Signatures

In this chapter, we initiate our discussion on the final problem in privacy-preserving computation that we address, which is related to the communication bandwidth and computational overhead of a privacy-preserving signature scheme, referred to as stealth signatures. In contrast to previous chapters, our primary focus here is on enhancing the *concrete* efficiency and bandwidth of the underlying scheme.

Existing stealth signature mechanisms either (1) exhibit security vulnerabilities in certain reasonable adversarial models or (2) demonstrate inefficiency in practical scenarios. In this chapter, we provide a formalization of *stealth signatures* through game-based definitions. We then introduce Spirit, the first efficient post-quantum secure stealth signature construction based on NIST standardized signature and key-encapsulation schemes, Dilithium and Kyber. The basic form of Spirit is secure only in a *weak* security model; however, we provide an efficiency-preserving and generic transform that *boosts* the security of Spirit to ensure the strongest security notion defined in this chapter. Compared to the state-of-the-art, our approach offers a ~ 3.37x improvement in signature size while maintaining signing and verification efficiency at around 0.2 ms.

We enhance Spirit by incorporating a *fuzzy tracking* functionality, allowing recipients to delegate the task of monitoring incoming transactions to a tracking server while upholding an anonymity notion similar to fuzzy message detection (FMD), which was recently introduced in [BLMG21]. Additionally, we extend Spirit with a novel fuzzy tracking framework called *scalable fuzzy tracking*, introduced in this chapter. This framework can be viewed as a counterpart to FMD, as it reduces the tracking server's computational workload to *sublinear* levels in the number of users, in contrast to FMD's linear workload. Experimental results demonstrate that, for millions of users, the server only requires 3.4 ms to filter each incoming message, representing at least a $\sim 76,000$x improvement over existing methods.

5.1 Overview

Cryptocurrency payments have revolutionized payment infrastructures by overcoming the need for a central authority and allowing for public verifiability.

On a very high level, cryptocurrency payments are made from a sender to a receiver, by posting a transaction onto a public ledger called *blockchain*. In the most basic form, the sender and receiver are identified by respective public keys (or addresses), and a transaction is authorized by the sender via a digital signature on the transaction wrt. the public key of the sender.

This paradigm allows users to make payments to any other user in the system without having to rely on banks, international money transfers- or exchange services. E-commerce [Webb], donation platforms [Webj, Webd, Webg], gaming platforms [Webc], etc., are just some of the popular use cases that are enabled by cryptocurrencies and their trustless payments. For example, donation platforms accept donations in the form of cryptocurrency payments, and to do this, a donation platform announces its addresses and users can make transactions paying to these addresses without requiring permission from any authority.

For transactions that want some level of anonymity, a critical weakness of the above paradigm is that it lacks reliable anonymity guarantees in its basic form. Several de-anonymization techniques [OKH13, SO13, RH13, RS13, MSH+17] have been demonstrated that link addresses on the blockchain to the real-world entities that own them. While de-anonymization of transactions may be beneficial in cases of preventing crimes such as money laundering, it has also led to questionable forms of censorship [Webe]. Combined with the public nature of blockchains, this also raises concerns about user privacy in general.

A mechanism known as *stealth addresses* [Webi, vS, Tod, CM17] was developed to address these anonymity issues. In the donation platform example above, the platform publishes a single master address, a so-called *stealth address*, and any user can send payments to the platform, by using a *locally re-randomized* version of the stealth address called one-time address. On one hand, such a one-time address is *unlinkable* to the stealth address for any outside observer, consequently, transactions to such a stealth address look as if they are going to random recipients (and not necessarily the donation platform). On the other hand, with access to its master secret, the donation platform can link such a one-time address to its stealth address and further generate the corresponding one-time secret locally, on the fly. Using this one-time secret, the coins associated with the one-time address can be spent.

The stealth address mechanism proposed in [vS] has in fact been deployed in many of the major currencies like Bitcoin [Webi], Ethereum [Webh], and Monero [vS]. The mechanism has further found direct application in privacy enhancement of payment protocols like Blitz [AMKM21].

As [vS] implements stealth addresses via signature schemes, we will refer to the cryptographic abstraction of the mechanism from [vS] as *stealth signatures*. Thus we will henceforth use the terms addresses and public keys interchangeably.

Notice that in the mechanism described above, the recipient only needs to publish its master address, and *does not* need to give out fresh unlinkable addresses for each potential sender. As the number of senders could well be in the hundreds or thousands (as is the case with e-commerce, donations, etc.), stealth signatures lead to a *scalable solution*.

Table 5.1: Comparison with Prior Works about Stealth Signatures

Works	w/KE[1]	Security	Post-quantum	opk Size	Signature Size
Monero's SS[vS]	○	sEUF-CMA	○	64 B	64 B
Paring-based SS[LYW+19]	●	EUF-CMA	○	231 B	115 B
ABB10-based SS[LLN+20]	●	EUF-CMA	●	3.35 GB	3.26 MB
[LLN+20] + NTRU (potential optimization)	●	EUF-CMA	●	13.82 KB	13.82 KB
Section A.2	◒[2]	sEUF-CMA	○	96 B	64 B
Section 5.5.1	○	EUF-CMA	●	2.08 KB	2.54 KB
Section 5.5.1+Dilithium (compiler from Section 5.4)	●	sEUF-CMA	●	2.08 KB	6.40 KB
Section 5.5.1+Falcon (compiler from Section 5.4)	●	sEUF-CMA	●	2.08 KB	4.09 KB

[1] Secure against key-exposures. Our construction presented in Section 5.5.1 can be upgraded to w/KE according to Section 5.4.
[2] Secure against bounded key-exposures.

Recent academic works [LYW+19, LLN+20] initiated the formal treatment of stealth signatures with cryptographic security guarantees. They observed that the construction of [vS] does not satisfy security under so-called *key-exposures*. Roughly, this means that if an adversary learns the corresponding one-time secret key for the one-time public keys that he generated, then he can potentially learn all one-time secret keys of all one-time public keys that he generates for this particular master address.

More recent proposals of stealth signature schemes [LYW+19, LLN+20] were designed to be secure against such key-exposure attacks, with the downside that their schemes use heavy tools such as pairings [BF01] or lattice basis delegation [ABB10]. These are currently not compatible with any of the major cryptocurrencies that exist today. Furthermore, with the threat of quantum computers looming large, cryptocurrency payments including the *pre-quantum* stealth signature mechanisms of [vS, LYW+19] remain vulnerable. While a lattice-based (and thus plausibly post-quantum) construction of stealth signatures was proposed in [LLN+20], this construction relies on the aforementioned lattice basis delegation. Consequently, their scheme is most likely too inefficient for practical use[1]. We compare our constructions and related works in Table 5.1. Please refer to Section 5.6 for more discussion.

this chapter is motivated by the following two questions:

- *Can we have an efficient stealth signature scheme with security against unbounded key-exposures, that is compatible with Schnorr, ECDSA and other group-based signature schemes predominantly used in currencies today?*

- *Can we have an efficient stealth signature scheme secure with unbounded key exposure that is post-quantum secure?*

A caveat of the stealth address mechanism described above is that a recipient (online or offline) has

[1]As the authors of [LLN+20] point out in Section 1.1, their "public key and signature sizes are too large for practical use".

Table 5.2: Comparison with Prior Works about Fuzzy/Private Tracking

Works	Privacy	Assumptions	Post-quantum	Server's Work	Latency/msg[2]	Receiver's Time
FMD_2[BLMG21]	ρN-anonymity[1]	Random Oracle	○	$O(N)$	933 sec	37.5 ms
Π_{TEE}[MSS+22]	Full Privacy	Trusted Execute Environment	○	$O(N)$	228 sec	12 ms
Π_{GC}[MSS+22]	Full Privacy	Two Non-colluding Servers	◒	$O(N)$	81.1 hour	1 ms
$\mathsf{OMR}_{\mathsf{p2}}$[LT22]	Full Privacy	Fully Homomorphic Encryption	●	$O(N)$	43.1 hour	63 ms
Section 5.5.2	ρN-anonymity	Standard Model	●	$O(N)$	11.70 sec	37.5 ms
Section 5.5.3	ρN-anonymity	Random Oracle	●	$O(\rho N)$	3.42 ms	37.5 ms

[1] ρ denotes the false-positive rate and N the number of clients.
[2] Calculated in a setting with $N = 2^{20}$ users and $M = 500,000$ messages based on the numbers from their papers. Latency per message induced by the server. See more discussion in Section 5.6.

to parse through a large number (hundreds of thousands per day) of transactions to identify those that send coins to one-time addresses corresponding to his master address.

A workaround was proposed in [vS], where a recipient can *delegate* identification of incoming payments to a semi-trusted third-party server called the *tracking server*. To do so, the recipient can generate a so-called *tracking key* from his secret key and provide it to the tracking server. The tracking key allows the tracking server to identify or *track* all incoming payments to the recipient using the tracking key, and later notify the recipient of these exact payments. On the other hand, such a tracking key should not enable the tracking server to generate one-time secrets for the concerned one-time addresses. Prior works [LYW+19, LLN+20] omit this tracking functionality in their formalization of stealth signatures.

A downside of the above tracking method is that the tracking server learns *exactly* which payments are addressed to the recipient, thus providing a tracking key to a server amounts to fully giving up on anonymity/unlinkability with respect to the tracking server. While there is a natural and obvious tension between the anonymity goal of unlinkability and the functional goal of trackability, a recent work by Beck et al. [BLMG21] attempts to strike a balance between these notions. They introduce the concept of *fuzzy message detection (FMD)*, where a tracking server can *approximately* detect messages meant for a recipient with an adjustable degree of uncertainty. More specifically, their notion of detection is *fuzzy* in the sense that messages meant for the recipient are always correctly identified, but there is a recipient-controlled false positive rate (baked into the *fuzzy* tracking key) which causes messages meant for other users to be misclassified as being meant for the recipient.

Thus, the tracking server cannot decide with certainty if a detected message is actually intended for the recipient or not. This mechanism makes it necessary for the sender of the message to include additional *fuzzy tracking information* and the tracking server possesses a fuzzy tracking key. Together, fuzzy tracking information and fuzzy tracking key enable fuzzy tracking. In principle, applying their technique to enable fuzzy tracking of one-time addresses in stealth signatures is fairly straightforward. However, relying on their schemes comes with considerable drawbacks. While their first scheme (FMD_2) is efficient, it relies on the pre-quantum DDH assumption. Their second scheme ($\mathsf{FMD}_{\mathsf{frac}}$) relies on garbled circuits as an additional component and is hence plausibly post-quantum. But the garbled circuits included in the ciphertext lead to an unacceptable size blowup of the sender's message. On the other hand, there are signalling detection or retrieval schemes [MSS$^+$22, LT22] for fully private tracking instead of fuzzy tracking, but all of them require linear work at the server side which doesn't scale to thousands or millions of users. We discuss their schemes and ours in Section 5.6 and present a comparison about this in Table 5.2.

This leads us to the following question:

- *Can we have a stealth signature scheme with efficient fuzzy tracking in the post-quantum setting and scalable to hundreds of thousands (or even millions) of users?*

5.1.1 Our Contributions

We summarize our contributions below.

Clear Constructions. We introduce Spirit (in Section 5.5.1), the first practically efficient post-quantum secure stealth signature scheme secure without key exposure. Towards this goal, we consider the Dilithium [LDK$^+$20] signature scheme which is lattice-based and a winner of the NIST competition. Without changing the signature scheme in any way, we augment Dilithium with additional algorithms to obtain Spirit so that it now supports one-time key derivations and tracking. One of the main motivations for considering Dilithium is that we believe it is one of the most popular and most likely post-quantum signature schemes to be adopted into cryptocurrencies for the authentication of payments.

Next, we show how one can generically transform (in Section 5.4) a stealth signature scheme that is secure *without* key-exposure into a scheme that is secure *with* unbounded key-exposure.

Thus we can transform Spirit into one that is practically efficient and secure with an unbounded key exposure. Both Spirit and the transformed construction are compatible with cryptocurrencies that would support Dilithium signature verification. Moreover, we do not require any additional support from the scripting language of cryptocurrency.

Furthermore, we construct a stealth signature scheme (in Section A.2) that is compatible with group-based schemes like Schnorr and ECDSA which are used in most of the currencies today. However, it only guarantees security with *bounded* key-exposure: It tolerates a-priori number of one-time secret key leakage.

Fuzzy Constructions. We then present two fuzzy stealth signature schemes (using Spirit), both of which are the first efficient and post-quantum candidates.

In the first construction (in Section 5.5.2), we take a similar approach as FMD from [BLMG21]. But we reduce its overhead from $O(\lambda)$ to 1 bit per signal by making novel use of ciphertext compression techniques [BDGM19]. Additionally, we show how to allow *finer* false-positive rates without requiring garbled circuits as in [BLMG21].

We then present a new *scalable* framework for fuzzy tracking (in Section 5.3.4) followed by an efficient construction (in Section 5.5.3) in the random oracle model. This framework can be viewed as a 'dual' version of the FMD mechanism from [BLMG21].

Intuitively, it is a trade-off between efficiency and usability: By limiting the users' ability to choose false-positive rates, we are able to reduce the tracking server's computational work to *sublinear* in the total number of users. This compares very favourably with prior works, where the server needs to take a linear scan of each user's tracking key [BLMG21, MSS+22, LT22].

Implementation. We implemented Spirit, post-quantum FMD, and scalable fuzzy tracking based on Dilithium, Kyber, and Falcon with anonymized open-source code [Weba]. We test them with different parameter sets on an ordinary laptop as presented in Table 5.3 and Table 5.4 (in Section 5.6). Experiment results show that our stealth signature with the strongest security only yields a 4.09 KB signature ($\sim$ 797x improvement), meanwhile, the verification time is less than 0.2 ms. Similarly, our scalable fuzzy tracking mechanism only takes 3.42 ms ($\sim$ 76,000x improvement) to filter each incoming message with millions of users in the setting.

5.2 Techniques

Let us first recall the group-based stealth signature scheme of [vS]: Given a cryptographic group $\mathbb{G} = \langle g \rangle$ of prime order p, the master public key is $\mathsf{mpk} := (g, h_0 := g^a, h_1 := g^b) \in \mathbb{G}^3$, where $\mathsf{msk} := (a \leftarrow\$ \mathbb{Z}_p, b \leftarrow\$ \mathbb{Z}_p)$ is the master secret key, and $\mathsf{mtk} := a$ is the tracking key. To re-randomize mpk to a one-time address (i.e., one-time public key), the sender samples a uniformly random $r \leftarrow\$ \mathbb{Z}_p$ and computes $\mathsf{opk} := g^{\mathsf{H}(h_0^r)} \cdot h_1 \in \mathbb{G}$ where $\mathsf{H} : \mathbb{G} \mapsto \mathbb{Z}_p$ is a hash function modelled as a random oracle. Additionally, the sender attaches tracking information $\mathsf{tki} := g^r \in \mathbb{G}$ to the opk. To derive the corresponding one-time secret key osk from msk, the receiver computes $\mathsf{osk} := \mathsf{H}(\mathsf{tki}^a) + b \in \mathbb{Z}_p$ with the help of tki. Now, the receiver can sign (for e.g., Schnorr or ECDSA) any message with osk to output a signature which can be verified with corresponding opk because of the discrete-log relation $\mathsf{opk} = g^{\mathsf{osk}}$. An additional mechanism is that $\mathsf{mtk} := a$ can be given to a tracking server for tracking: By comparing whether $\mathsf{opk} \overset{?}{=} g^{\mathsf{H}(\mathsf{tki}^{\mathsf{mtk}})} \cdot h_1$, the tracking server can determine whether opk links to the issuer of mtk.

Taking a closer look, this approach to build a stealth signature apparently can be generically decomposed to a linearly homomorphic one-way function $\mathsf{f} : \mathcal{D} \mapsto \mathcal{M}$ where $\mathsf{f}(x+y) = \mathsf{f}(x) + \mathsf{f}(y)$, and a key-exchange protocol $(\mathsf{KE}_1, \mathsf{KE}_2, \mathsf{KE}_3)$, where KE_i denotes the i-th message function:

$$\begin{aligned} \mathsf{ct}_1 \in \mathcal{C}_1 &\leftarrow \mathsf{KE}_1(r_1), \\ (\mathsf{ct}_2 \in \mathcal{C}_2, K \in \mathcal{K}) &\leftarrow \mathsf{KE}_2(r_2, \mathsf{ct}_1), \\ K \in \mathcal{K} &\leftarrow \mathsf{KE}_3(r_1, \mathsf{ct}_2), \end{aligned}$$

where r_1, r_2 are two user's secrets, and K is the agreed-upon key. Here $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{K}$ are the first message, the second message and the key space, respectively. Now, let $\mathsf{mpk} := \big(\mathsf{ct}_1 := \mathsf{KE}_1(r_1), B := \mathsf{f}(b)\big)$ and $\mathsf{msk} := (r_1, b), \mathsf{mtk} := r_1$. To publish a one-time address, the sender can just compute $(\mathsf{ct}_2, K) \leftarrow \mathsf{KE}_2(r_2, \mathsf{ct}_1)$ and publish

$$\mathsf{opk} := B + \mathsf{f}\big(\mathsf{H}(K)\big), \mathsf{tki} := \mathsf{ct}_2$$

where $\mathsf{H} : \mathcal{K} \mapsto \mathcal{D}$. Correspondingly,

$$\mathsf{osk} := b + \mathsf{H}\big(\mathsf{KE}_3(r_1, \mathsf{tki})\big).$$

Since they obey the relation $\mathsf{f}(\mathsf{osk}) = \mathsf{opk}$, we can leverage this to sign and verify. The tracking mechanism still works by checking if

$$\mathsf{opk} \overset{?}{=} \mathsf{f}\Big(\mathsf{H}\big(\mathsf{KE}_3(\mathsf{mtk}, \mathsf{tki})\big)\Big) + B.$$

We will now adopt this blueprint to construct a stealth signature in the lattice setting.

5.2.1 Spirit: Lattice-based Stealth Signature

To make our protocol both *efficient* and *practical*, we would like to use optimized NIST winners as our building blocks. In this chapter, we choose Dilithium as the underlying digital signature considering that it is one of the most popular signature schemes in NIST [LDK+20]. We call the resulting stealth signature scheme Spirit. Basically, it follows the above approach: In Dilithium, the public key is a Module Learning With Errors (MLWE) [BGV12] sample $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ where its secret-error pair $(\mathbf{s}_1, \mathbf{s}_2)$ (both chosen from a suitable *short* distribution) acts as the secret key. Since MLWE involves only linear operations, we have that

$$\mathbf{t} + \mathbf{t}' = \mathbf{A}(\mathbf{s}_1 + \mathbf{s}_1') + \mathbf{s}_2 + \mathbf{s}_2'.$$

Yet, even though adding samples is approximately linearly homomorphic, this addition will increase *error rates* or lengths for both $\mathbf{s}_1$ and $\mathbf{s}_2$. Typically, the $\mathbf{s}_1$ and $\mathbf{s}_2$ are generated by sampling their coefficients uniformly with absolute value at most η (for some small parameter η). The increased norm of the new secrets $(\mathbf{s}_1 + \mathbf{s}_1', \mathbf{s}_2 + \mathbf{s}_2')$ will incur additional running time during signing due to the so-called "Fiat-Shamir with Abort" mechanism of Dilithium. To alleviate this issue, we only prove Spirit to be *existential unforgeable*. This will give us better parameters to balance security and efficiency. Looking ahead, we point out that Spirit can be transformed to an *strongly existentially unforgeable* scheme using a generic compiler which we will introduce later.

Apart from linearly homomorphic one-way functions, we still need a key-exchange protocol. However, this key exchange needs some additional properties. Specifically, we need a non-interactive key exchange (NIKE) protocol which is *substantially* stronger than KE we depicted above. The starting point is that it needs to be *anonymous* under chosen plaintext attacks (CPA), which means giving the message ct_2, the adversary cannot link it to the ct_1 used to generate ct_2. This is for stealth signatures as we don't want our one-time address to be linkable to the original master public address. This security notion is formalized as *unlinkability*.

But anonymity under chosen plaintext attacks will not even suffice yet for our applications. We will require a stronger notion of anonymity under plaintext checking attacks (PCA). Here, the adversary is given an additional oracle which allows him to check whether a ciphertext-plaintext pair is valid or not. To see why this is necessary, consider an adversary who is trying to link some $(\mathsf{opk}, \mathsf{tki})$ to

mpk. Such an adversary will be *able* to sample $\mathsf{ct}_2 \leftarrow\!\!\$\ \mathcal{C}_2, K \leftarrow\!\!\$\ \mathcal{K}$ to generate $(\mathsf{opk}', \mathsf{tki}')$, which can then be published to see if the tracking check passes. It turns out that anonymity under plaintext checking attacks is sufficient for this setting. However, we currently don't have a simple construction satisfying anonymity under plaintext checking attacks. As a consequence, we use an even stronger key-exchange protocol which is anonymous under chosen ciphertext attacks (CCA), namely, it is ANO-CCA-secure (formalized in Definition 1.1.5). Fortunately, the recent standardized KEM by NIST, Kyber [SAB+20], can be slightly modified to be ANO-CCA-secure [GMP22] and we use Kyber in the concrete instantiation. There are multiple technical details not covered in this outline, for instance, besides tki, the one-time address opk itself also needs to be anonymous. We refer to Section 5.5.1 for detailed construction and analysis.

So far, we briefly mentioned two important security notions for stealth signatures, namely unforgeability and unlinkability (Section 5.3 for formalization). However, we note that we only formalize these two notions as unforgeability *without* key-exposure and unlinkability *without* key-exposure, respectively. It turns out the above approach to build stealth signatures (as well as in Spirit) is *no longer secure* if a one-time secret key osk leaks: Suppose the sender learns osk somehow, he can instantly recover msk as

$$b := \mathsf{osk} - \mathsf{H}\big(\mathsf{KE}_3(r_1, \mathsf{tki})\big),$$

if he knows r_1 which is used to generate corresponding opk.

5.2.2 Generic Transformation: Security with Key-exposure

As mentioned above and noticed in prior works[LYW+19, NMRL16], leaking one-time secret keys is almost as bad as leaking the master secret key. This is a potential issue in current practical stealth signature schemes [vS] and it is costly to avoid. For instance, if we are willing to use techniques implying hierarchical identity-based encryption (HIBE), we could have a stealth signature scheme secure *with* key-exposure attacks by using pairing[BF01, LYW+19], lattice basis delegation[ABB10, LLN+20], or non-black box tools[DG17b]. All of above techniques are several orders of magnitude slower in computational time, or orders of magnitude larger in the signature or one-time public key size.

The reason we don't have a simple solution to this issue is that one-time secret keys are usually a linear function of the msk as mentioned in [LRR+19]. Apparently, we can achieve security with *bounded* key-exposure by adding more secrets in msk where bounded key-exposure means msk remains secure if the number of leaked osk is smaller than some 'a priori bound' and we show a candidate construction in Section A.2. However, any generic-group-based techniques to prevent unbounded key exposure should imply IBE which is known to be impossible using only black-box techniques [PRV12, SGS21].

In this chapter, we provide a conceptually simple, generic, and powerful black-box compiler to tackle this problem in the context of stealth signatures (in Section 5.4): We use a short chain of signatures[Mer90] to compile any stealth signature $\mathsf{SS}_{\mathsf{w/o}}$ secure *without* key-exposure into a strong stealth signature SS_{w} secure *with* unbounded key-exposure. The high-level idea is to break this 'linear' relation between osk and msk. Specifically, instead of generating osk directly, with the help of an additional digital signature DS, we generate

$$\mathsf{osk} := (\sigma_1, \mathsf{sk}, \mathsf{vk}),$$

where $\sigma_1 \leftarrow \mathsf{SS}_{\mathsf{w/o}}.\mathsf{Sign}(\mathsf{osk}', \mathsf{vk})$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{DS}.\mathsf{Gen}(\lambda)$. Note that osk' is the one-time secret key in the scheme $\mathsf{SS}_{\mathsf{w/o}}$. Intuitively, since osk has a non-linear relation with msk, the adversary cannot recover msk from osk as $\mathsf{SS}_{\mathsf{w/o}}$ is unforgeable. To sign a message m, it runs $\sigma_2 \leftarrow \mathsf{DS}.\mathsf{Sign}(\mathsf{sk}, m)$ and outputs the final signature $\sigma := (\sigma_1, \sigma_2, \mathsf{vk})$. Similarly, to verify σ just use opk to verify the signature σ_1 on vk and use vk to verify the signature σ_2 on m. Compared to the original stealth signature $\mathsf{SS}_{\mathsf{w/o}}$, our compiled one SS_{w} incurs a slightly larger signature size and longer verification time, but in turn, is far more efficient than the above HIBE-related techniques.

Additionally, we show this compiler can also leverage $\mathsf{SS}_{\mathsf{w/o}}$ with existential unforgeability to SS_{w} with strong unforgeability via a small tweak: Instead of signing on m, we sign as $\sigma_2 \leftarrow \mathsf{DS}.\mathsf{Sign}(\mathsf{sk}, m||\sigma_1)$. This prevents strong unforgeability attacks of SS_{w} because: Assuming vk in σ is not altered, a different $\sigma_1' \neq \sigma_1$ will lead to a forgery $(m||\sigma_1', \sigma_2)$ of DS in SS_{w}. Therefore, Spirit can also be leveraged in this way to be strongly unforgeable with a key exposure. This gives us the first practical post-quantum SS_{w} secure with a key exposure.

5.2.3 Fuzzy Tracking

We will now turn to the issue that in the above constructions, the tracking mechanism will leak the users' metadata to the tracking servers, i.e., the tracking server will know exactly which mtk belongs to which specific $(\mathsf{opk}, \mathsf{tki})$. As discussed above, to address this problem, Beck et al. [BLMG21] proposed a mechanism named fuzzy message detection (FMD): The server is given a fuzzy tracking key ftk instead of mtk to *filter* incoming fuzzy tracking information ftki for its users. Here, ftki is attached with $(\mathsf{opk}, \mathsf{tki})$. Specifically, for unmatched ftki and ftk, they will be linked with probability roughly ρ.

Transforming their scheme to a post-quantum world is non-trivial as there are still two potential obstacles in the lattice setting: First, it is not practically efficient since its ftki is as large as $O(n \cdot |\mathsf{ct}|)$-bit where $|\mathsf{ct}| = \mathsf{poly}(\lambda)$. This is highly undesirable in practice as our expectation is something like $O(\lambda) + n$. The other problem is the uniformly-ambiguous (recalled in Chapter 1) encryption, as it is unclear how to extend the random oracle-based approach in [BLMG21], to the lattice setting due to the presence of noise. We show that these two obstacles are related and can be resolved simultaneously. For simplicity, assume $n = 1$ for the moment. Recall that in Regev encryption with modulus q, the ciphertext is composed of two parts, a vector $\mathbf{c}_1 \in \mathbb{Z}_q^\ell$ and a scalar $c_2 \in \mathbb{Z}_q$. The secret key is $\mathbf{s} \in \mathbb{Z}_q^\ell$ and decryption consists of rounding after a linear operation:

$$\lceil \mathbf{s}^T \mathbf{c}_1 - c_2 \rfloor_2 = \lceil \frac{q}{2} \cdot m + e \rfloor_2,$$

where $e < B < \frac{q}{4}$ is a bounded error. This is not just bad for efficiency (as we need additional $n \log q$ bits to encrypt n more bits), but also for security: With the correct secret key $\mathbf{s}$, $\mathbf{s}^T\mathbf{c}_1 - c_2$ is distributed as a Gaussian around $\frac{q}{2}$ or 0; With a wrong key $\mathbf{s}_*$, $\mathbf{s}_*^T\mathbf{c}_1 - c_2$ is distributed uniformly random over the entire domain $\mathbb{Z}_q$. These two cases are clearly distinguishable by an adversary.

Our solution will be to compress c_2 into a single bit, which doesn't convey enough information about the distribution. Hence this idea will solve both of the above problems simultaneously. Brakerski et al. [BDGM19] introduced *rate-1* packed Regev encryption which can compress each c_2 to just one bit but require an additional offset scalar $z \in \mathbb{Z}_q$ in the header. Thus to to encrypt n bits, the ciphertext after compression is $(\mathbf{c}_1, z, w_1, \ldots, w_n)$ where $w_i \in \{0,1\}$. To make the offset z statistically close to uniformly random (in our setting pseudorandom doesn't suffice because the adversary gets the secret key), we require *super-polynomial* noise-modulus ratio of Learning With Errors (LWE)[Rego5] which makes the scheme slightly less efficient. This gives us a lattice-based fuzzy tracking scheme (and ambiguous encryption), and surprisingly, it doesn't rely on heuristic assumptions like random oracles which are necessary in [BLMG21].

5.2.4 Scalable Fuzzy Tracking

We observe that in the above FMD style tracking, the server's computational work is $O(N)$ with N users and is not scalable when thousands (or millions) of users are using the service of the server. We provide a framework for *scalable* fuzzy tracking which we view as a dual version of FMD [BLMG21], where the server's work is *sublinear*. In this framework, we weaken the requirement that the false-positive rate can be adaptively changed by users. Instead, it is fixed in advance in this setting. This weakening is reasonable as it was shown in [SPB21] that an adversary can mount statistical attacks if users have varying false positive rates. To circumvent such attacks it was suggested that all users have high enough false positivity rates as even a small subset of low-rate users can affect unlinkability for the entire pool of users. Therefore we can fix the false positivity rate to be a high enough value for everyone. For example, as calculated in [SPB21], the false-positive rate ρ is better to be as large as $\frac{1}{\sqrt{N}}$. In this case, we can make the server's overhead $O(\rho N)$ for each incoming message which was at least $O(N)$ in prior works [BLMG21, MSS$^+$22, LT22].

We let the tracking server run FTKGen in the beginning to publish fuzzy public key fpk and secretly hold the fuzzy tracking key ftk. For each ftki received from senders, the tracking server will expand ftki to a list of size t composed of potential users' master public keys to which ftki may belong. The tracking server can then store $(\mathsf{opk}, \mathsf{tki})$ in the mailbox of each candidate in this list. Crucially, the master public keys of other potential candidates should remain uncontrollable to either the sender or the server. Otherwise, the sender might manipulate the chance of each key appearing in the list. This additional property is named *unbiasedness*. This rules out the trivial solution, where for instance the sender just sends directly a range of master public keys including the targeted mpk.

Since mpk of each user can be large, in our construction we hash $\mathsf{mpk} \in \mathcal{K}$ to some small $\mathsf{hint} \in \mathcal{T}$ (while making $|\mathcal{T}| \geq N$) and use the hint to locate each user's mailbox. Our scheme is based on the underlying IND-CPA encryption of Kyber, except that we use a non-prime modulus. For instance, assuming the hint contains $n = \lceil \log N \rceil$ bits, i.e., $b := \mathsf{hint} \in \{0,1\}^n$, to generate ftki, the sender modifies the Kyber512's ciphertext $\mathsf{ct} := (\mathbf{c}_1, c_2)$ to $\mathsf{ct}' := (\mathbf{c}_1', c_2')$ as follows:

$$\mathbf{c}_1' := \mathbf{c}_1 + \frac{q}{2}\begin{bmatrix} x_i \\ 0 \end{bmatrix} \;\; c_2' := c_2 + \frac{q}{2} y_i,$$

where ct (and ct') encrypts hint_i as the plaintext, $x_i \leftarrow \mathsf{encode}_{R_q}(\mathbf{x}_i)$ is a polynomial mapped from the vector $\mathbf{x}_i$, and $\mathbf{x}_i, \mathbf{y}_i \in \{0,1\}^m \leftarrow \mathsf{H}(\delta, i)$ are outputs of a hash function H with the seed δ. Here $i \in [t]$ denotes the i-th target mpk as the intended recipient.

For $\mathsf{ftki} := \mathsf{ct}'$, the tracking server decrypts ct' using the key $\mathsf{sk} = \mathbf{s}$ as follows: for $\forall j \in [t]$,

$$\mathsf{hint}_j \leftarrow \mathsf{decode}_{R_q}(\lceil \mathbf{s}^T(\mathbf{c}_1' - \frac{q}{2}\begin{bmatrix} x_j \\ 0 \end{bmatrix}) - c_2' \rfloor_2 \oplus y_j),$$

to get t potential hints. To argue privacy, intuitively, since $\mathbf{s}$ remains random to the sender, the decrypted hint for $j \neq i$ would also be random to the sender as

$$\mathsf{hint}_j = \mathsf{hint}_i \oplus (y_j \oplus y_i) \oplus \mathsf{decode}_{R_q}(\lceil \frac{q}{2}\mathbf{s}^T \begin{bmatrix} x_j \\ 0 \end{bmatrix} \rfloor_2).$$

However, to prove the unbiasedness we mentioned above, we need to be careful because the standard regularity lemma seems hard to apply with such a small noise parameter and modulus in ideal lattices. Our solution is to rely on the specific structure of the corresponding cyclotomic polynomial and shows that even $\mathbf{s}^T \begin{bmatrix} x_j \\ 0 \end{bmatrix}$ is not close to a uniformly random polynomial but there's enough entropy to make hint_j uniformly random over $\{0,1\}^n$ as long as n is much smaller than the degree of the polynomial.

5.3 Definitions

In this section we first present our formal definitions for a stealth signature scheme, followed by how we can add-on fuzziness to the scheme. Note that stealth signatures were formalized in prior works [LYW+19, LLN+20], however our formalization of security is strictly stronger than theirs, and moreover we are the first to formalize tracking and fuzzy tracking for a stealth signature scheme. We will point out the exact differences in the formalism as we introduce the security notions formally.

Below we present the definition of stealth signatures, that formalizes the tracking of keys which was absent in prior works. This formalization allows for tracking to be outsourced to third-party servers.

Definition 5.3.1. A *stealth signature* (SS) scheme consists of the PPT algorithms (MKGen, OPKGen, OSKGen, Track, Sign, Vf) that are defined as follows.

$\underline{(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk}) \leftarrow \mathsf{MKGen}(\lambda)}$: the master key generation algorithm takes as input the security parameter λ and outputs the master public key mpk, the master secret key msk, and the master tracking key mtk.

$\underline{(\mathsf{opk}, \mathsf{tki}) \leftarrow \mathsf{OPKGen}(\mathsf{mpk})}$: the one-time public key generation algorithm takes as input the master public key mpk and outputs the one-time public key opk and a tracking information tki.

$\underline{\mathsf{osk}/\perp \leftarrow \mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})}$: the one-time secret key generation algorithm takes as input the master secret key msk, the one-time public key opk, and the tracking information tki, and outputs a one-time secret key osk or a special symbol $\perp$.

$\underline{\mathsf{true}/\mathsf{false} \leftarrow \mathsf{Track}(\mathsf{mtk}, \mathsf{opk}, \mathsf{tki})}$: the tracking algorithm takes as input the master tracking key mtk, the one-time public key opk, and the tracking information tki, and outputs true or false.

$\underline{\sigma/\perp \leftarrow \mathsf{Sign}(\mathsf{osk}, m)}$: the signing algorithm takes as input the one-time secret key osk, and a message m, and outputs a signature σ or a special symbol $\perp$.

$\underline{\mathsf{true}/\mathsf{false} \leftarrow \mathsf{Vf}(\mathsf{opk}, m, \sigma)}$: the verification algorithm takes as input the one-time public key opk, a message m, and a signature σ, and outputs true or false.

The notion of correctness if formalized below.

Definition 5.3.2 (Correctness). A SS scheme $(\mathsf{MKGen}, \mathsf{OPKGen}, \mathsf{OSKGen}, \mathsf{Track}, \mathsf{Sign}, \mathsf{Vf})$ is said to be *correct* if for all $\lambda \in \mathbb{N}$, all $(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk}) \leftarrow \mathsf{MKGen}(\lambda)$, all $(\mathsf{opk}, \mathsf{tki}) \leftarrow \mathsf{OPKGen}(\mathsf{mpk})$, all $\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})$, we have the following that hold simultaneously:

- we have $\Pr[\mathsf{Track}(\mathsf{mtk}, \mathsf{opk}, \mathsf{tki}) = \mathsf{true}] = 1$
- we have $\Pr[\mathsf{Vf}(\mathsf{opk}, m, \mathsf{Sign}(\mathsf{osk}, m)) = \mathsf{true}] = 1$,

note that sometimes we don't require *perfect* correctness and having correctness probability $1-\mathsf{negl}(\lambda)$ instead would suffice.

5.3.1 Security of SS Without Key Exposure

In terms of security, we first want unforgeability, which guarantees that it is infeasible for an adversary to forge a signature on a (fresh) message wrt. some one-time public key opk^* for a master public key mpk. The adversary is given access to a one-time secret key generation oracle $\mathsf{OSKGen}\mathcal{O}$ using which the adversary can generate a fresh one-time secret key. However, the adversary does not get to learn the generated one-time secret keys, therefore the notion is said to be *without key exposure*. The adversary also has access to a signing oracle, to which it can query a signature on any message of its choice wrt. any one-time secret key that has been generated with a query to $\mathsf{OSKGen}\mathcal{O}$. The formal definition is presented below.

Definition 5.3.3 (Unforgeability without key-exposure). A SS scheme $(\mathsf{MKGen}, \mathsf{OPKGen}, \mathsf{OSKGen}, \mathsf{Track}, \mathsf{Sign}, \mathsf{Vf})$ is said to be *unforgeable without key exposure* if there exists a negligible function negl for all $\lambda \in \mathbb{N}$, and for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\text{sEUF-CMA}^{\mathcal{A}}_{\mathsf{w/o-ke}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\text{sEUF-CMA}_{\mathsf{w/o-ke}}$ is defined in Figure 5.1.

```
$\text{EUF-CMA}^{\mathcal{A}}_{\mathsf{w/o-ke}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk}) \leftarrow \mathsf{MKGen}(\lambda)$
$\mathsf{OK} := [], Q := \emptyset$
$(m^*, \sigma^*, i^*)$
    $\leftarrow \mathcal{A}^{\mathsf{OSKGen}\mathcal{O}, \mathsf{Sign}\mathcal{O}}(\mathsf{mpk}, \mathsf{mtk})$
$(\mathsf{opk}^*, \mathsf{osk}^*) := \mathsf{OK}[i^*]$
$b_0 := (m^*, \cdot, i^*) \notin Q$
  $/ (m^*, \sigma^*, i^*) \notin Q$ for $\text{sEUF-CMA}_{\mathsf{w/o-ke}}$
$b_1 := \mathsf{Vf}(\mathsf{opk}^*, m^*, \sigma^*) \stackrel{?}{=} \mathsf{true}$
**return** $b_0 \wedge b_1$

$\mathsf{OSKGen}\mathcal{O}(\mathsf{opk}, \mathsf{tki})$
$\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})$
$\mathsf{OK} := \mathsf{OK} || (\mathsf{opk}, \mathsf{osk})$
**return** 1

$\mathsf{Sign}\mathcal{O}(i, m)$
$(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{OK}[i]$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{osk}, m)$
$Q := Q \cup (m, \sigma, i)$
**return** $\sigma$
```

Figure 5.1: Experiment for unforgeability without key exposure.

We then want unlinkability, which guarantees that it is infeasible for an adversary to associate a one-time public key to the master public key wrt. which it was generated. The adversary is given two master public keys mpk_0 and mpk_1, while also given a challenge one-time public key opk_b and the corresponding tracking information tki_b (for $b \in \{0, 1\}$) generated wrt. mpk_b. The adversary is given access to the $\mathsf{OSKGen}\mathcal{O}$ as before, and a signing oracle. The adversary is not given access to any of the one-time secret keys and therefore the notion is said to be *without key exposure*. The formal definition is presented below.

Definition 5.3.4 (Unlinkability without key-exposure)**.** A SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf) is said to be *unlinkability without key exposure* if there exists a negligible function negl for all $\lambda \in \mathbb{N}$, and for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\text{UNLNK}^{\mathcal{A}}_{\mathsf{w/o-ke}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\text{UNLNK}_{\mathsf{w/o-ke}}$ is defined in Figure 5.2.

5.3.2 Security of SS With Key Exposure

Prior works [LYW+19, LLN+20] formalized security with additionally giving adversary the one-time secret keys, i.e., the $\mathsf{OSKGen}\mathcal{O}$ returns the generated osk to the adversary.

The unforgeability notion *with key exposure* is formalized below. Notice that our formalization exposes the one-time secret keys osk to the adversary except the key wrt. which the adversary forges the signature.

Definition 5.3.5 (Unforgeability with key-exposure)**.** A SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf) is said to be *unforgeable with key exposure* if there exists a negligible function negl for

```
$\mathrm{UNLNK}^{\mathcal{A}}_{\mathsf{w/o-ke}}(\lambda)$
$(\mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{mtk}_0) \leftarrow \mathsf{MKGen}(\lambda)$
$(\mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{mtk}_1) \leftarrow \mathsf{MKGen}(\lambda)$
$\mathsf{OK}_0 := \mathsf{OK}_1 := []$
$b \leftarrow \{0, 1\}$
$(\mathsf{opk}_b, \mathsf{tki}_b) \leftarrow \mathsf{OPKGen}(\mathsf{mpk}_b)$
$\mathsf{osk}_b \leftarrow \mathsf{OSKGen}(\mathsf{msk}_b, \mathsf{opk}_b, \mathsf{tki}_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{OSKGen}\mathcal{O}, \mathsf{Sign}\mathcal{O}}(X, \mathsf{opk}_b, \mathsf{tki}_b)$
    / where $X := (\mathsf{mpk}_0, \mathsf{mpk}_1)$
$b_0 := (b = b')$
return $b_0$

$\mathsf{OSKGen}\mathcal{O}(b^*, \mathsf{opk}, \mathsf{tki})$
$\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}_{b^*}, \mathsf{opk}, \mathsf{tki})$
$\mathsf{OK}_{b^*} := \mathsf{OK}_{b^*} || (\mathsf{opk}, \mathsf{osk})$
return $1$

$\mathsf{Sign}\mathcal{O}(b^*, i, m)$
if $i = -1$ then
    $\sigma \leftarrow \mathsf{Sign}(\mathsf{osk}_b, m)$
else
    $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{OK}_{b^*}[i]$
    $\sigma \leftarrow \mathsf{Sign}(\mathsf{osk}, m)$
return $\sigma$
```

Figure 5.2: Experiment for unlinkability without key exposure.

```
$\mathrm{sEUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{w-ke}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk}) \leftarrow \mathsf{MKGen}(\lambda)$
$\mathsf{OK} := [], Q := \emptyset$
$(m^*, \sigma^*, i^*)$
    $\leftarrow \mathcal{A}^{\mathsf{OSKGen}\mathcal{O}, \mathsf{Sign}\mathcal{O}}(\mathsf{mpk}, \mathsf{mtk})$
$(\mathsf{opk}^*, \mathsf{osk}^*, \cdot) := \mathsf{OK}[i^*]$
$b_0 := (m^*, \sigma^*, i^*) \notin Q$
$b_1 := \mathsf{Vf}(\mathsf{opk}^*, m^*, \sigma^*) = 1$
$b_2 := (\mathsf{OK}[i^*] \neq (\cdot, \cdot, \mathsf{true}))$
return $b_0 \wedge b_1 \wedge b_2$

$\mathsf{OSKGen}\mathcal{O}(i, \mathsf{opk}, \mathsf{tki}, \mathsf{flag})$
if $\mathsf{OK}[i] = (\mathsf{opk}, \cdot, \cdot) \wedge \mathsf{flag} = \mathsf{true}$
return $\mathsf{OK}[i].\mathsf{osk}$
$\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})$
$\mathsf{OK} := \mathsf{OK} || (\mathsf{opk}, \mathsf{osk}, \mathsf{flag})$
if $\mathsf{flag} = \mathsf{true}$ then return $\mathsf{osk}$
else return $1$

$\mathsf{Sign}\mathcal{O}(i, m)$
$(\mathsf{opk}, \mathsf{osk}, \mathsf{flag}) \leftarrow \mathsf{OK}[i]$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{osk}, m)$
$Q := Q \cup (m, \sigma, i)$
return $\sigma$
```

Figure 5.3: Experiment for unforgeability with key exposure.

all $\lambda \in \mathbb{N}$, and for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\mathrm{sEUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{w-ke}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathrm{sEUF\text{-}CMA}_{\mathsf{w/o-ke}}$ is defined in Figure 5.3.

The notion of unlinkability *with key exposure* is formalized below. Similar to the case above, the $\mathsf{OSKGen}\mathcal{O}$ returns the generated osk. Our formalization apart from the tracking functionality is stronger than prior works in that the adversary is even given the challenge one-time secret key osk_b.

```
$\mathrm{UNLNK}^{\mathcal{A}}_{\mathsf{w-ke}}(\lambda)$
$(\mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{mtk}_0) \leftarrow \mathsf{MKGen}(\lambda)$
$(\mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{mtk}_1) \leftarrow \mathsf{MKGen}(\lambda)$
$\mathsf{OK}_0 := \mathsf{OK}_1 := []$
$b \leftarrow \{0,1\}$
$(\mathsf{opk}_b, \mathsf{tki}_b) \leftarrow \mathsf{OPKGen}(\mathsf{mpk}_b)$
$\mathsf{osk}_b \leftarrow \mathsf{OSKGen}(\mathsf{msk}_b, \mathsf{opk}_b, \mathsf{tki}_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{OSKGen}\mathcal{O}}(X, \mathsf{opk}_b, \mathsf{tki}_b, \mathsf{osk}_b)$
    / where $X := (\mathsf{mpk}_0, \mathsf{mpk}_1)$
$b_0 := (b = b')$
return $b_0$

$\mathsf{OSKGen}\mathcal{O}(b^*, \mathsf{opk}, \mathsf{tki})$
$\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}_{b^*}, \mathsf{opk}, \mathsf{tki})$
$\mathsf{OK}_{b^*} := \mathsf{OK}_{b^*} || (\mathsf{opk}, \mathsf{osk})$
return $\mathsf{osk}$
```

Figure 5.4: Experiment for unlinkability with key exposure.

Definition 5.3.6 (Unlinkability with key-exposure)**.** A SS scheme $(\mathsf{MKGen}, \mathsf{OPKGen}, \mathsf{OSKGen}, \mathsf{Track}, \mathsf{Sign}, \mathsf{Vf})$ is said to be *unlinkability with key exposure* if there exists a negligible function negl for all $\lambda \in \mathbb{N}$, and for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\mathrm{UNLNK}^{\mathcal{A}}_{\mathsf{w-ke}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathrm{UNLNK}_{\mathsf{w/o-ke}}$ is defined in Figure 5.4.

5.3.3 Fuzzy Stealth Signatures

We now formally incorporate the fuzzy tracking functionality into the definition of stealth signing.

Definition 5.3.7 (Fuzzy Stealth Signatures)**.** A *fuzzy stealth signatures* (F-SS) scheme is a SS scheme $(\mathsf{MKGen}, \mathsf{OPKGen}, \mathsf{OSKGen}, \mathsf{Track}, \mathsf{Sign}, \mathsf{Vf})$ with additional interfaces $(\mathsf{FTKGen}, \mathsf{FTrack})$ defined below.

$\underline{(\mathsf{opk}, \mathsf{tki}, \mathsf{ftki}) \leftarrow \mathsf{OPKGen}(\mathsf{mpk})}$: overloading the interface OPKGen to output the fuzzy tracking information ftki.

$\underline{\mathsf{ftk} \leftarrow \mathsf{FTKGen}(\mathsf{mtk}, \rho)}$: the fuzzy tracking key generation algorithm takes as input the master tracking key mtk, and a false positivity rate ρ, and outputs a fuzzy tracking key ftk.

$\underline{\mathsf{true}/\mathsf{false} \leftarrow \mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki})}$: the fuzzy tracking algorithm takes as input the fuzzy tracking key ftk, the fuzzy tracking information ftki, and outputs true or false.

We define the notion of correctness below. We borrow the notion of fuzziness from [BLMG21] and adapt the same for the stealth signature setting. Intuitively, the correctness of fuzzy tracking says that with a probability ρ, the fuzzy tracking algorithm returns true for a mismatched fuzzy tracking key and a one-time public key. For a correctly matched fuzzy tracking key and a one-time public key, the tracking algorithm always returns true.

Definition 5.3.8 (Correctness for fuzzy tracking)**.** An F-SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTKGen, FTrack) is said to be *correct* if the original SS scheme is correct and if for all $\lambda \in \mathbb{N}$, all $\rho \in (0,1]$, all $(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk}) \leftarrow \mathsf{MKGen}(\lambda)$, all $(\mathsf{opk}, \mathsf{tki}, \mathsf{ftki}) \leftarrow \mathsf{OPKGen}(\mathsf{mpk})$, all $\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})$, all $\mathsf{ftk} \leftarrow \mathsf{FTKGen}(\mathsf{mtk}, \rho)$, we have the following that holds simultaneously:

- $\Pr[\mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki}) = \mathsf{true}] = 1$
- and for any $\mathsf{ftki}' \notin \mathsf{SUPP}(\mathsf{OPKGen}(\mathsf{mpk}))$, we have

$$\Pr\big[\mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki}') = \mathsf{true}\big] = \rho.$$

The unforgeability notion is the same as in Figure 5.3 as the adversary in the notion is given the master tracking key already.

Unlinkability with fuzzy tracking guarantees that it is infeasible for an adversary given two fuzzy tracking keys, both of which return a true or a false when tracking a challenge one-time public key $(\mathsf{opk}_b, \mathsf{ftki}_b)$ simultaneously, to associate $(\mathsf{opk}_b, \mathsf{ftki}_b)$ with the correct tracking key, i.e., either ftk_0 or ftk_1. The adversary is said to violate the notion if it can guess correctly the association non-negligibly more than 1/2.

Definition 5.3.9 (Unlinkability with key-exposure and fuzzy tracking)**.** A F-SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTKGen, FTrack) is said to be *unlinkable with key-exposure and fuzzy tracking* if there exists a negligible function negl for all $\lambda \in \mathbb{N}$, all $\rho \in (0,1]$, and for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\big[\mathrm{UNLNK}^{\mathcal{A}}_{\mathsf{fw-ke}}(\lambda, \rho) = 1\big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathrm{UNLNK}_{\mathsf{fw-ke}}$ is defined in Figure 5.5.

5.3.4 Scalable Fuzzy Tracking

We now formalize the functionality, correctness, and security of fuzzy scalable stealth signatures as follows.

Definition 5.3.10 (Fuzzy Scalable Stealth Signatures)**.** A *fuzzy scalable stealth signature* (F-SSS) is a SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf) with additional interfaces (FTKGen, FTrack) and a modified OPKGen defined below.

$\underline{(\mathsf{fpk}, \mathsf{ftk}) \leftarrow \mathsf{FTKGen}(\rho, N)}$: the fuzzy tracking key generation algorithm takes as input a false positivity rate ρ, and the number of total users N, and outputs a fuzzy tracking key ftk and fuzzy public key fpk. The algorithm is run by the tracking server ahead of time.

```
$\mathrm{UNLNK}_{\mathsf{fw-ke}}(\lambda)$
$\mathsf{OK}_0 := \mathsf{OK}_1 := []$
$(\mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{mtk}_0) \leftarrow \mathsf{MKGen}(\lambda)$
$(\mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{mtk}_1) \leftarrow \mathsf{MKGen}(\lambda)$
$b \leftarrow \{0,1\}$
$(\mathsf{opk}_b, \mathsf{tki}_b, \mathsf{ftki}_b) \leftarrow \mathsf{OPKGen}(\mathsf{mpk}_b)$
$\mathsf{osk}_b \leftarrow \mathsf{OSKGen}(\mathsf{msk}_b, \mathsf{opk}_b, \mathsf{tki}_b)$
$(\mathsf{st}_{\mathcal{A}}, \rho) \leftarrow \mathcal{A}_1(\mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{opk}_b,$
            $\mathsf{tki}_b, \mathsf{ftki}_b, \mathsf{osk}_b)$
$\mathsf{ftk}_0 \leftarrow \mathsf{FTKGen}(\mathsf{mtk}_0, \rho)$
$\mathsf{ftk}_1 \leftarrow \mathsf{FTKGen}(\mathsf{mtk}_1, \rho)$
$b_1 \leftarrow \mathsf{FTrack}(\mathsf{ftk}_0, \mathsf{ftki}_b)$
$b_2 \leftarrow \mathsf{FTrack}(\mathsf{ftk}_1, \mathsf{ftki}_b)$
if $b_1 = b_2$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{OSKGen}\mathcal{O}}(\mathsf{st}_{\mathcal{A}}, \mathsf{ftk}_0, \mathsf{ftk}_1)$
else
  $b' \leftarrow\$ \{0,1\}$
return $(b = b')$

$\mathsf{OSKGen}\mathcal{O}(b^*, \mathsf{opk}, \mathsf{tki})$
$\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}_{b^*}, \mathsf{opk}, \mathsf{tki})$
$\mathsf{OK}_{b^*} := \mathsf{OK}_{b^*} || (\mathsf{opk}, \mathsf{osk})$
return $\mathsf{osk}$
```

Figure 5.5: Experiment for unlinkability of F-SS with key-exposure.

$\underline{(\mathsf{opk}, \mathsf{tki}, \mathsf{ftki}) \leftarrow \mathsf{OPKGen}(\mathsf{mpk}, \mathsf{fpk})}$: overloading the interface OPKGen to additionally take input fpk and output fuzzy tracking information ftki.

$\underline{\mathsf{list} \leftarrow \mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki})}$: the fuzzy tracking algorithm takes as input the fuzzy tracking key ftk, the fuzzy tracking information ftki, and outputs a list consisting of master public keys.

Definition 5.3.11 (Correctness for fuzzy scalable stealth signatures). An F-SSS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTKGen, FTrack) is said to be *correct* if the original SS scheme is correct and if for all $\lambda \in \mathbb{N}$, all $\rho \in (0,1]$, all $(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk}) \leftarrow \mathsf{MKGen}(\lambda)$, all $(\mathsf{opk}, \mathsf{tki}, \mathsf{ftki}) \leftarrow \mathsf{OPKGen}(\mathsf{mpk}, \mathsf{fpk})$, all $\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})$, all $(\mathsf{fpk}, \mathsf{ftk}) \leftarrow \mathsf{FTKGen}(\rho, N)$, we have the following that holds simultaneously:

- $\Pr[\mathsf{mpk} \in \mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki})] = 1$
- and for any $\mathsf{mpk}' \neq \mathsf{mpk}$, we have

$$\Pr\left[\mathsf{mpk}' \in \mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki})\right] \approx \rho.$$

Crucially, we omit opk in FTrack as ftki is already associated with opk and we still have the regular Track algorithm that works with tk, opk and tki for tracking. The correctness definition above 'ties' together the keys ftk, mpk and mtk, and $(\mathsf{opk}, \mathsf{tki}, \mathsf{ftki}) \leftarrow \mathsf{OPKGen}(\mathsf{mpk})$ by requiring that $\mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki})$ always returns 1.

```
$\mathrm{UNLNK}_{\mathsf{fs}_{\mathsf{w-ke}}}(\lambda, \rho, N)$
$\mathsf{OK}_0 := \mathsf{OK}_1 := []$
$(\mathsf{mpk}_0, \mathsf{msk}_0, \mathsf{mtk}_0) \leftarrow \mathsf{MKGen}(\lambda)$
$(\mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{mtk}_1) \leftarrow \mathsf{MKGen}(\lambda)$
$(\mathsf{fpk}, \mathsf{ftk}) \leftarrow \mathsf{FTKGen}(\rho, N)$
$b \leftarrow \{0,1\}$
$(\mathsf{opk}_b, \mathsf{tki}_b, \mathsf{ftki}_b) \leftarrow \mathsf{OPKGen}(\mathsf{mpk}_b,$
                                        $\mathsf{fpk})$
$\mathsf{osk}_b \leftarrow \mathsf{OSKGen}(\mathsf{msk}_b, \mathsf{opk}_b, \mathsf{tki}_b)$
$\mathsf{list} \leftarrow \mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki}_b)$
**if** $\mathsf{mpk}_0 \in \mathsf{list} \wedge \mathsf{mpk}_1 \in \mathsf{list}$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{OSKGen}\mathcal{O}}(\mathsf{ftk}, \mathsf{mpk}_0, \mathsf{mpk}_1,$
          $\mathsf{opk}_b, \mathsf{tki}_b, \mathsf{ftki}_b, \mathsf{osk}_b)$
**else**
  $b' \leftarrow\!\$\ \{0,1\}$
$b_0 := (b = b')$
**return** $b_0$
```

```
$\mathrm{UNI\text{-}UBS}_{\mathsf{fs}}(\lambda, \rho, N)$
$(\mathsf{fpk}, \mathsf{ftk}) \leftarrow \mathsf{FTKGen}(\rho, N)$
$(\mathsf{st}_{\mathcal{A}}, \mathsf{ftki}, i, j, \mathsf{mpk}) \leftarrow \mathcal{A}_1(\mathsf{fpk})$
$\mathsf{list} \leftarrow \mathsf{FTrack}(\mathsf{ftk}, \mathsf{ftki})$
$b \leftarrow\!\$\ \{0,1\}$
**if** $\mathsf{list}[i] \neq \mathsf{mpk} \vee i = j \vee \mathsf{mpk} \neq \mathcal{K}$
  $b' \leftarrow\!\$\ \{0,1\}$
**else**
  $\mathbf{v}^0 := \mathsf{list}[j], \mathbf{v}^1 \leftarrow\!\$\ \mathcal{K}$
  $b' \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \mathbf{v}^b)$
**return** $b \overset{?}{=} b'$
```

```
$\mathsf{OSKGen}\mathcal{O}(b^*, \mathsf{opk}, \mathsf{tki})$
$\mathsf{osk} \leftarrow \mathsf{OSKGen}(\mathsf{msk}_{b^*}, \mathsf{opk}, \mathsf{tki})$
$\mathsf{OK}_{b^*} := \mathsf{OK}_{b^*} || (\mathsf{opk}, \mathsf{osk})$
**return** $\mathsf{osk}$
```

Figure 5.6: Experiments for unlinkability and uniformly unbiasedness of F-SSS with Key-Exposure.

Definition 5.3.12 (Unlinkability with key-exposure and fuzzy scalable tracking). A F-SSS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTKGen, FTrack) is said to be *unlinkable with key-exposure and fuzzy scalable tracking* if there exists a negligible function negl for all $\lambda \in \mathbb{N}$, all $\rho \in (0,1]$, and for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\mathrm{UNLNK}^{\mathcal{A}}_{\mathsf{fs}_{\mathsf{w-ke}}}(\lambda, \rho, N) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathrm{UNLNK}_{\mathsf{fs}_{\mathsf{w-ke}}}$ is defined in Figure 5.6. Note that, similar to prior works, we only consider the semi-honest server in the definition.

Definition 5.3.13 (Unbiasedness for fuzzy scalable tracking). A F-SSS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTKGen, FTrack) is said to be *unbiased by senders* if there exists a negligible function negl, for all $\lambda \in \mathbb{N}$, and for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\mathrm{UNI\text{-}UBS}^{\mathcal{A}}_{\mathsf{fs}}(\lambda, \rho, N) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the experiment $\mathrm{UNI\text{-}UBS}_{\mathsf{fs}}$ is defined in Figure 5.6 where $\mathsf{list}[i]$ denotes the i-th item of the list and $\mathcal{K}$ denotes the master public key space.

```
MKGen($\lambda$)
    return $\mathsf{SS}_{w/o}.\mathsf{MKGen}(\lambda)$

OSKGen(msk, opk, tki)
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{DS.Gen}(\lambda)$
    $\mathsf{epk} \leftarrow \mathsf{SS}_{w/o}.\mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})$
    return $\perp$ if $\mathsf{epk} = \perp$
    $\sigma_1 \leftarrow \mathsf{SS}_{w/o}.\mathsf{Sign}(\mathsf{epk}, \mathsf{vk})$
    return $\mathsf{osk} := (\sigma_1, \mathsf{sk}, \mathsf{vk})$

Track(mtk, opk, tki)
    return $\mathsf{SS}_{w/o}.\mathsf{Track}(\mathsf{mtk}, \mathsf{opk}, \mathsf{tki})$

OPKGen(mpk)
    return $\mathsf{SS}_{w/o}.\mathsf{OPKGen}(\mathsf{mpk})$

Sign(osk, $m$)
    return $\perp$ if $\mathsf{osk} = \perp$
    $(\sigma_1, \mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{osk}$
    $\sigma_2 \leftarrow \mathsf{DS.Sign}(\mathsf{sk}, m || \sigma_1)$
    return $\sigma := (\sigma_1, \sigma_2, \mathsf{vk})$

Vf(opk, $\sigma$, $m$)
    $(\sigma_1, \sigma_2, \mathsf{vk}) := \sigma$
    if $\mathsf{SS}_{w/o}.\mathsf{Vf}(\mathsf{opk}, \sigma_1, \mathsf{vk}) \wedge$
        $\mathsf{DS.Vf}(\mathsf{vk}, \sigma_2, m || \sigma_1)$
      return 1
    else return 0
```

Figure 5.7: A generic transformation to lift $\mathsf{SS}_{w/o}$ to SS_{w}.

5.4 Generic Transformation of Stealth Signatures

We provide our black-box compiler below to upgrade an $\mathsf{SS}_{w/o}$ without key-exposure to an SS_{w} with key-exposure.

Suppose we have a digital signature scheme DS which is strongly unforgeable sEUF-CMA. Then we have a black-box compiler leveraging SS to stronger version as shown in Figure 5.7. Basically, the compiler transforms any $\mathsf{SS}_{w/o}$ with EUF-CMA$_{w/o-ke}$ and UNLNK$_{w/o-ke}$ security (without key-exposure) into an SS_{w} with sEUF-CMA$_{w-ke}$ and UNLNK$_{w-ke}$ security (with key-exposure).

It is easy to see that correctness always holds as long as $\mathsf{SS}_{w/o}$ and DS are correct. The security of unforgeability and unlinkability for SS_{w} are captured informally in the following theorem. The formal theorem and security proofs are deferred to Section 5.7.

Theorem 5.4.1 (informal). *The stealth signature* SS_{w} *constructed in Section 5.4 is secure in sEUF-CMA*$_{w-ke}$ *and UNLNK*$_{w-ke}$ *experiments if* $\mathsf{SS}_{w/o}$ *is EUF-CMA*$_{w/o-ke}$ *secure, UNLNK*$_{w/o-ke}$ *secure, and* DS *is sEUF-CMA secure.*

5.5 Spirit: Lattice-based (Fuzzy) Stealth Signature

We first describe Spirit and later show we can make it fuzzy.

```
MKGen(λ)
    $\mathbf{A} \in R_q^{k\times\ell} \leftarrow \mathsf{Dil.ExpandA}(\mathsf{crs})$
    $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow\!\!\$\ S_\eta^\ell \times S_\eta^k$
    $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
    $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KEM.Gen}(\lambda)$
    $\mathsf{mpk} := (\mathbf{t}, \mathsf{ek}),$
    $\mathsf{msk} := (\mathbf{s}_1, \mathbf{s}_2, \mathsf{dk}, \mathbf{t})$
    $\mathsf{mtk} := (\mathsf{dk}, \mathbf{t})$
    return $(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk})$

OSKGen(msk, opk, tki)
    $(\mathbf{s}_1, \mathbf{s}_2, \mathsf{dk}, \mathbf{t}) := \mathsf{msk}$
    if $\mathsf{false} \leftarrow \mathsf{Track}((\mathsf{dk}, \mathbf{t}), \mathsf{opk}, \mathsf{tki})$
        return $\perp$
    $K \leftarrow \mathsf{KEM.Decaps}(\mathsf{dk}, \mathsf{tki})$
    $(\mathbf{s}'_1, \mathbf{s}'_2) \leftarrow \mathsf{Dil.ExpandS}(K)$
    return $\mathsf{osk} := (\mathbf{s}_1 + \mathbf{s}'_1, \mathbf{s}_2 + \mathbf{s}'_2)$

Vf(opk, σ, m)
    return $\mathsf{Dil.Vf}(\mathsf{opk}, \sigma, m)$

OPKGen(mpk)
    $(\mathbf{t}, \mathsf{ek}) := \mathsf{mpk}$
    $\mathbf{A} \leftarrow \mathsf{Dil.ExpandA}(\mathsf{crs})$
    $(C, K) \leftarrow \mathsf{KEM.Encaps}(\mathsf{ek})$
    $(\mathbf{s}'_1, \mathbf{s}'_2) \in S_\eta^\ell \times S_\eta^k \leftarrow \mathsf{Dil.ExpandS}(K)$
    $\mathbf{t}' := \mathbf{t} + \mathbf{A}\mathbf{s}'_1 + \mathbf{s}'_2$
    $(\mathbf{t}'_1, \cdot) \leftarrow \mathsf{Dil.power2Round}(\mathbf{t}', d)$
    return $(\mathsf{opk} := \mathbf{t}'_1, \mathsf{tki} := C)$

Sign(osk, m)
    return $\perp$ if $\mathsf{osk} = \perp$
    return $\sigma := \mathsf{Dil.Sign}(\mathsf{osk}, m)$

Track(mtk, opk, tki)
    $(\mathsf{dk}, \mathbf{t}) := \mathsf{mtk}$
    $\mathbf{A} \leftarrow \mathsf{Dil.ExpandA}(\mathsf{crs})$
    $K \leftarrow \mathsf{KEM.Decaps}(\mathsf{dk}, \mathsf{tki})$
    $(\mathbf{s}'_1, \mathbf{s}'_2) \leftarrow \mathsf{Dil.ExpandS}(K)$
    $\tilde{\mathbf{t}} := \mathbf{t} + \mathbf{A}\mathbf{s}'_1 + \mathbf{s}'_2$
    $(\tilde{\mathbf{t}}_1, \cdot) \leftarrow \mathsf{Dil.power2Round}(\tilde{\mathbf{t}}, d)$
    return $\mathsf{opk} \overset{?}{=} \tilde{\mathbf{t}}_1$
```

Figure 5.8: Construction of Spirit with EUF-CMA$_{\mathsf{w/o-ke}}$ and UNLNK$_{\mathsf{w/o-ke}}$ security

5.5.1 Lattice-based Stealth Signature

We use an ANO-CCA-secure key exchange KEM (Kyber) [SAB+20] and an EUF-CMA-secure signature (Dilithium) to construct an SS scheme with *existential unforgeability without key-exposure* and *unlinkability without key-exposure* in random oracle model. We require a common reference string $\mathsf{crs} \leftarrow\!\!\$\ \{0,1\}^{256}$, but for conciseness, we omit the explicit mention of crs in interfaces. We provide the detailed construction in Figure 5.8.

Intuitively, we use KEM to re-randomize the underlying master secret key msk to obtain osk each time and it needs to be actively anonymous which can be instantiated by Kyber with slight modification as shown in [GMP22]. Also, we only require Dilithium to be EUF-CMA secure which gives us a larger space to choose parameters. We recall Dilithium as follows.

Definition 5.5.1 (Dilithium [LDK+20])**.** Dilithium denoted by Dil is a post-quantum digital signature DS scheme based on the "Fiat-Shamir with Aborts" approach [Lyu09, Lyu12]. It is based on MLWE, MSIS and SelfTargetMSIS assumptions with ring $R_q := \mathbb{Z}_q[X]/(X^m + 1)$. Moreover, for secrets $\mathbf{s} \leftarrow\!\!\$\ S_\eta^\ell$, each coefficient of the vector is an element of R_q with small coefficients of size at most

η. In its optimized construction, there are some useful supporting algorithms which we described as follows:

- $\mathsf{ExpandA}(\mathsf{crs})$: The function maps a uniform seed crs to a matrix $\mathbf{A} \in R_q^{k\times\ell}$.
- $\mathsf{ExpandS}(K)$: The function used for generating the secret vectors in key generation, maps a seed K to $(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta^\ell \times S_\eta^k$.
- $\mathsf{power2Round}(r, d)$: The function is the straightforward bit-wise way to break up an element $r := r_1 \cdot 2^d + r_0$ where $r_0 = r \mod 2^d$ and $r_1 = (r - r_0)/2^d$.
- $\mathsf{HighBits}_q(r, \alpha)$: The function select an α that is a divisor of $q-1$ and write $r = r_1 \cdot \alpha + r_0$ in the same way as before then returns r_1.
- $\mathsf{MakeHint}_q(z, r, \alpha)$: The function runs $r_1 \leftarrow \mathsf{HighBits}(r, \alpha)$ and $v_1 \leftarrow \mathsf{HighBits}(r+z, \alpha)$, then returns $r_1 \neq v_1$.

Correctness. Since

$$\mathbf{t}' = \mathbf{t} + \mathbf{A}\mathbf{s}_1' + \mathbf{s}_2' = \mathbf{A}(\mathbf{s}_1' + \mathbf{s}_1) + (\mathbf{s}_2 + \mathbf{s}_2'),$$

it is easy to see we have $1 - \mathsf{negl}(\lambda)$ correctness as long as underlying KEM and Dil have $1 - \mathsf{negl}(\lambda)$ correctness.

Notably, $\mathbf{s}_1' + \mathbf{s}_1$ and $\mathbf{s}_2 + \mathbf{s}_2'$ have roughly doubled the norm thus doubling β in signatures. This will incur additional iterations in Sign as the number of repetitions is roughly $2^{-256\cdot\beta(\frac{\ell}{\gamma_1}+\frac{k}{\gamma_2})}$ where $\gamma_1 \approx 2\gamma_2$[LDK+20]. However, besides having doubled β, we can increase γ_1 and γ_2 to $2\gamma_1$ and $2\gamma_2$, respectively. This tweak just slightly lowers the $\mathsf{SelfTargetMSIS}$ hardness but won't harm the running time. To see this, in Dil's proof, the reduction's advantage mainly dominated by $\mathsf{MSIS}_{k,\ell,4\gamma_2}$ for sEUF-CMA security, but $\mathsf{SelfTargetMSIS}_{k,\ell+1,2\gamma_2}$ for EUF-CMA security. Without using the forking lemma (since it is not tight and not possible in a quantum setting), the hardness of $\mathsf{SelfTargetMSIS}$ is mainly from finding some sort ($\|\cdot\|_\infty \leq 2\gamma_2$) vectors $\mathbf{z}, \mathbf{u}'$ such that $\mathbf{A}\mathbf{z} + \mathbf{u}' = \mathbf{t}'$ and amounts to MSIS problem (referring to Section 6.2.1 and Appendix C.3 in [LDK+20] for details). Therefore, doubled γ_2 in our SPIRIT construction gives the reduction of EUF-CMA$_{\mathsf{w/o-ke}}$ roughly the same advantage as that of sEUF-CMA in Dil. We present the concrete security level in Table 5.3.

Security Analysis. We prove the construction of SPIRIT in Figure 5.8 is existential unforgeable and unlinkable *without* key exposure, and is secure in EUF-CMA$_{\mathsf{w/o-ke}}$ and UNLNK$_{\mathsf{w/o-ke}}$ experiment, respectively. For the security of EUF-CMA$_{\mathsf{w/o-ke}}$, we prove this in two steps. First, we show it is unforgeable without key exposure under no-message attacks (NMA), i.e., the adversary cannot query $\mathsf{Sign}\mathcal{O}(\cdot)$, and we refer the corresponding experiment to UF-NMA$_{\mathsf{w/o-ke}}$; Next, we show a reduction from UF-NMA$_{\mathsf{w/o-ke}}$ to EUF-CMA$_{\mathsf{w/o-ke}}$. Since Dil does not rely on the lower parts of the public key $\mathbf{t}_0$ to be secret, so for simplicity, we assume the one-time public key opk is $\mathbf{t}'$ instead of $\mathbf{t}_1'$. Also, we assume $\mathsf{crs} := \mathbf{A}$ directly and is publicly known.

Lemma 5.5.1 (informal). *SPIRIT in Figure 5.8 is unforgeable without key exposure under no-message attacks if* $\mathsf{SelfTargetMSIS}$ *and* MLWE *assumptions hold.*

```
MKGen(λ, n)
    mpk′, mtk′, msk′ ← SS.MKGen(λ)
    (pk, sk) ← pRgv.Gen(λ, n)
    mpk := (mpk′, pk), msk := msk′
    mtk := (mtk′, sk)
    return (mpk, mtk, msk)

FTKGen(mtk, ρ)
    parse (mtk′, sk) := mtk
    parse (s_1, ..., s_n) := sk
    t ← ⌊log_2(1/ρ)⌉
    return ftk := (s_1, ..., s_t)

Sign(osk, m)
    return SS.Sign(osk, m)

Vf(opk, σ, m)
    return SS.Vf(opk, σ, m)

OPKGen(mpk)
    parse (mpk′, pk) := mpk
    ftki ← pRgv.Enc(pk, 1)
    return (SS.OPKGen(mpk′), ftki)

FTrack(ftk, ftki)
    [m_1, ..., m_|ftk|] ← pRgv.Dec(ftk, ftki)
    b := ⋀_{i=1}^{|ftk|} m_i
    return b =? 1

Track(mtk, opk, tki)
    parse (mtk′, sk) := mtk
    return SS.Track(mtk′, opk, tki)

OSKGen(msk, opk, tki)
    return SS.OSKGen(msk, opk, tki)
```

Figure 5.9: Post-quantum FMD fuzzy tracking

Then we have the following theorems to show the construction is unforgeable and unlinkable. The formal statement and analysis of the above lemma and the following theorem is deferred to Section 5.7.1.

Theorem 5.5.2 (informal). *Spirit in Figure 5.8 is existential unforgeable and unlinkable without key exposures if it is UF-NMA$_{\mathsf{w/o-ke}}$ and the KEM used is ANO-CCA secure.*

5.5.2 Lattice-based Fuzzy Stealth Signature

We provide a lattice-based construction for fuzzy tracking in a standard model. Basically, it is packed Regev encryption with ciphertext compression [BDGM19]. And this gives us the first post-quantumly ambiguous encryption without relying on random oracles.

Packed Regev (compressed). We first recall the construction of packed Regev with ciphertext compression [BDGM19] in Figure 5.13 of Section 5.7.2, where χ is the error distribution and B is the error bound between $z + c_{2,i}$ and $z + \mathbf{s}_i^T \mathbf{c}_1$.

Note that apart from the header $(\mathbf{c}_1, z)$, the payload (w_i) are just n bits which is almost as succinct as DLog-based fuzzy message detection scheme FMD$_2$ in [BLMG21]. Specifically, the entire ciphertext is $(\ell + 1) \log q + n$-bit large.

Since IND-CPA and IK-CPA security (recalled in Chapter 1) of pRgv are discussed in prior works already, we focus on its ambiguous security and we show it is actually Uniformly-Ambiguous (recalled in Chapter 1) with super-poly noise-modulus ratio. The formal statement and proof of the lemma below are deferred to Section 5.7.2.

Lemma 5.5.3. *Packed Regev encryption* pRgv *with ciphertext compression shown in Figure 5.13 satisfies Definition 1.1.6 and is uniformly-ambiguous UNI-AMB-secure when* $\frac{4Bn}{q}$ *is* $\mathsf{negl}(\lambda)$.

The modulus of Figure 5.13. To argue uniformly-ambiguous security, we need a super-polynomial noise-to-modulus ratio (e.g., 60-bit modulus in our case) which is usually assumed in homomorphic encryption-related works. This is a somewhat stronger assumption since it assumes the lattice problem BDD or GapSVP is hard to even with super-polynomial approximation factor [Rego5].

Construction. We then provide a lattice-based fuzzy stealth signature in Figure 5.9, which is composed of a standard stealth signature SS and a compressed packed Regev encryption pRgv shown above. Basically, it uses the same framework as FMD_1 presented in [BLMG21].

Correctness. We provide the correctness analysis in Section 5.7.2.

Now we consider the false-positive rate ρ when using different fuzzy tracking keys. Since $\mathbf{c}_1$ looks uniformly random due to LWE assumption, $\mathbf{s}_i^T \mathbf{c}_1$ is uniformly random over $\mathbb{Z}_q$ by Leftover Hash Lemma as the inner product is a strong randomness extractor. This implies $\lceil \mathbf{s}_i^T \mathbf{c}_1 + z \rfloor$ is uniformly random over $\{0, 1\}$ and FTrack returns true with probability $2^{-t} \approx \rho$.

Security Analysis. Formal statement and corresponding proof of the following theorem are deferred to Section 5.7.2

Theorem 5.5.4 (informal). *The fuzzy stealth signature constructed in Figure 5.9 is* unlinkable with key-exposure and fuzzy tracking *if the underlying stealth signature is* $UNLNK_{\mathsf{w-ke}}$ *and* pRgv *is UNI-AMB and IK-CPA secure.*

We also provide an approach to extend it to a finer false-positive rate as shown in Section 5.7.2.

5.5.3 Scalable Lattice-based Fuzzy Tracking

As discussed in Section 5.2.4, we limit the user's ability to choose a false-positive rate and provide a new framework of fuzzy tracking which is substantially more scalable than prior works[BLMG21, MSS+22, LT22]. Please refer to Section 5.3.4 for functionality and security definitions.

Construction. We describe the detailed construction in Figure 5.10, where $\{0, 1\}^{2m} \leftarrow \mathsf{H}(k \in \{0, 1\}^{\lambda}, i \in [t])$ is a hash function with the seed k and $\mathsf{H}_n : \{0, 1\}^{|\mathsf{mpk}|} \mapsto \{0, 1\}^n$ is another hash function mapping mpk to a hint which is used to locate mpk's mailbox in server's storage. Since it is based on Module-LWE assumption, R_q denotes the ring $Z_q[X]/(X^m+1)$, and $\mathsf{encode}_{R_q} : \{0, 1\}^m \mapsto Z_q[X]/(X^m + 1)$ is a function mapping binary strings to the ring elements with binary coefficients; Similarly, decode_{R_q} is the reverse operation to map back to a binary string. Basically, it is a variant

```
MKGen($\lambda$)
return SS.MKGen($\lambda$)

FTKGen($\rho, N$)
$n := \lceil \log_2 N \rceil$
$t := \lceil \rho \cdot 2^n \rceil$
$\mathbf{A} \in R_q^{\ell \times \ell} \leftarrow\!\!\$\ \mathsf{crs}$
$(\mathbf{s}, \mathbf{e}) \leftarrow\!\!\$\ (B_\eta^\ell)^2$
$\mathbf{b} := \mathbf{A}\mathbf{s} + \mathbf{e}$
return ftk := $(\mathbf{s}, t)$, fpk := $(\mathbf{b}, \mathsf{H}_n, t)$

FTrack(ftk, ftki)
parse $(\mathbf{s}, t)$ := ftk, $(\mathbf{c}_1, c_2, \delta)$ := ftki
$\forall i \in [t]$ :
    $(\mathbf{x}^i, \mathbf{y}^i) \leftarrow \mathsf{H}(\delta, i)$
$x^i, y^i \in R_q \leftarrow \mathsf{encode}_{R_q}(\mathbf{x}^i, \mathbf{y}^i)$
    $\mathbf{c}_1^i := \mathbf{c}_1 - \frac{q}{2} \cdot (\begin{bmatrix} x^i \\ 0 \end{bmatrix})$
    $w^i := \lceil \mathbf{s}^T \mathbf{c}_1^i - c_2 \rfloor_2 \oplus y^i$
$\mathsf{hint}^i := \mathsf{decode}_{R_q}(w^i)[: n]$
return list := $\{\mathsf{hint}^1, \ldots, \mathsf{hint}^t\}$

Sign(osk, $m$)
return SS.Sign(osk, $m$)
```

```
OPKGen(mpk, fpk)
parse $(\mathbf{b} \in R_q^\ell, \mathsf{H}_n, t)$ := fpk
hint $\in \{0,1\}^n \leftarrow \mathsf{H}_n(\mathsf{mpk})$
$\mathbf{z} \leftarrow\!\!\$\ \{0,1\}^{m-n}$
$\mathbf{w}^T := [\mathsf{hint}^T \| \mathbf{z}^T]$
$i \leftarrow\!\!\$\ [t]$
$\delta \leftarrow\!\!\$\ \{0,1\}^\lambda$
$(\mathbf{r}, \mathbf{e}_1) \leftarrow\!\!\$\ (B_\eta^\ell)^2, e_2 \leftarrow\!\!\$\ B_\eta$
$(\mathbf{x}, \mathbf{y} \in \{0,1\}^m) \leftarrow \mathsf{H}(\delta, i)$
$x, y, w \in R_q \leftarrow \mathsf{encode}_{R_q}(\mathbf{x}, \mathbf{y}, \mathbf{w})$
$\mathbf{c}_1 := \mathbf{A}^T \mathbf{r} + \mathbf{e}_1 + \frac{q}{2} \cdot \begin{bmatrix} x \\ 0 \end{bmatrix}$
$c_2 := \mathbf{b}^T \mathbf{r} + e_2 + \frac{q}{2} \cdot (w + y)$
ftki := $(\mathbf{c}_1, c_2, \delta)$
return SS.OPKGen(mpk), ftki

OSKGen(msk, opk, tki)
return SS.OSKGen(msk, opk, tki)

Track(mtk, opk, tki)
return SS.Track(mtk, opk, tki)

Vf(opk, $\sigma$, $m$)
return SS.Vf(opk, $\sigma$, $m$)
```

Figure 5.10: Scalable lattice-based fuzzy tracking

of the underlying IND-CPA encryption of Kyber with a non-prime modulus. Though we lose the advantage of NTT multiplications, we can still mitigate this by using Karatsuba and Toom-Cook algorithms.

Correctness. It is clear to see that the targeted mpk must have $\mathsf{hint} := \mathsf{hint}^i = \mathsf{H}_n(\mathsf{mpk})$ appears in list with probability 1: For the targeted index $i \in [t]$, we have $\mathbf{c}_1^i = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ which is the same as standard ciphertext header. The decryption will output hint directly as long as $q > 4B$. Now we focus on the other case where $\mathsf{mpk}' \neq \mathsf{mpk}$. Firstly, considering $\mathsf{hint}_j \in \mathsf{list}$, it is decrypted as

$$\lceil \mathbf{s}^T \mathbf{c}_1^j - c_2 \rfloor_2 \oplus y^j = \lceil e' + \frac{q}{2}(w + s_1(x^i - x^j) + y^i) \rfloor_2 \oplus y^j,$$

where s_1 is the first ring element of $\mathbf{s}$. hint_j is uniformly random over $\{0,1\}^n$ after rounding $\lceil \cdot \rfloor_2$ as

Table 5.3: Performance Result of Constructions in Section 5.4 and Section 5.5.1

Scheme[1]	w/KE	sec	sec_q[2]	opk	Signature	MKGen	OPKGen	Track	OSKGen	Sign	Vf
Spirit2	○	114	104	2.08 KB	2.54 KB	0.068 ms	0.074 ms	0.076 ms	0.078 ms	0.208 ms	0.053 ms
Spirit3	○	171	155	3.04 KB	3.45 KB	0.131 ms	0.137 ms	0.136 ms	0.138 ms	0.377 ms	0.089 ms
Spirit5	○	245	223	4.16 KB	4.81 KB	0.191 ms	0.198 ms	0.202 ms	0.215 ms	0.443 ms	0.145 ms
Dilithium2+Spirit2	●	114	104	2.08 KB	6.40 KB	0.070 ms	0.076 ms	0.078 ms	0.358 ms	0.222 ms	0.114 ms
Dilithium3+Spirit3	●	171	155	3.04 KB	8.85 KB	0.129 ms	0.136 ms	0.132 ms	0.597 ms	0.369 ms	0.182 ms
Dilithium5+Spirit5	●	245	223	4.16 KB	12.2 KB	0.186 ms	0.193 ms	0.197 ms	0.762 ms	0.428 ms	0.291 ms
Falcon512+Spirit2	●	114	104	2.08 KB	4.09 KB	0.069 ms	0.074 ms	0.075 ms	5.458 ms	0.226 ms	0.074 ms
Falcon1024+Spirit3	●	171	155	3.04 KB	6.51 KB	0.133 ms	0.133 ms	0.133 ms	17.7 ms	0.444 ms	0.130 ms
Falcon1024+Spirit5	●	245	223	4.16 KB	7.88 KB	0.194 ms	0.198 ms	0.201 ms	17.5 ms	0.441 ms	0.185 ms

[1] Spirit2 builds on Dilithium2 and anonymized Kyber512, Spirit3 builds on Dilithium3 and anonymized Kyber768, and Spirit5 builds on Dilithium5 and anonymized Kyber1024.
[2] sec_q indicates the hardness of Quantum Core-SVP whereas sec indicates that of Classical Core-SVP.

$y^j \oplus y^i$ are outputs of the random oracle H. Then, for any $\mathsf{mpk}' \neq \mathsf{mpk}$, $\Pr[\mathsf{H}_n(\mathsf{mpk}') = \mathsf{hint}_j] = \frac{1}{2^n}$ since H_n is a random oracle, and

$$\begin{aligned}\Pr[\mathsf{H}_n(\mathsf{mpk}') \in \mathsf{list}] &= \sum_{j=1}^{t} \Pr[\mathsf{H}_n(\mathsf{mpk}') = \mathsf{hint}_j] \\ &= \frac{t}{2^n \approx N} \approx \rho.\end{aligned}$$

Security Analysis. The formal theorem statements and proof of the following theorems are deferred to Section 5.7.3.

Theorem 5.5.5 (informal). *The fuzzy scalable stealth signature constructed in Figure 5.10 is unlinkable with key exposure and fuzzy tracking if the underlying stealth signature is UNLNK$_{\mathsf{w-ke}}$ and* MLWE *holds. It is also* unbiased *and satisfying UNI-UBS$_{\mathsf{fs}}$ defined in Definition 5.3.13 if $n \leq \frac{m}{2}$ where m is a power of 2 and B_η is a centered binomial distribution.*

5.6 Performance Analysis

5.6.1 Implementations

We present the performance result in Table 5.3 and Table 5.4. The open source code can be checked at [Weba].

For Spirit in Section 5.5.1, similar to Dilithium, we denote the scheme with three security levels as Spirit2, Spirit3, and Spirit5. Parameters are the same as Dilithium's, except that our $\beta, \gamma_1, \gamma_2$ are doubled. Moreover, we use a variant of Kyber in Spirit: Replacing the original FO transform of Kyber with the one suggested in [GMP22] which makes Kyber ANO-CCA-secure. For Post-quantum

Table 5.4: Performance Result of Constructions in Section 5.5.2 and Section 5.5.3

Scheme	sec	sec_q	N Clients	ρ	Public Key	Fuzzy Tracking Info	Setup Time	OPKGen[1]	FTrack[2]
Post-quantum FMD	104	94	2^{20}	2^{-10}	345.6 KB	17.2 KB	108.8 ms	75.13 ms	11.74 sec
Post-quantum FMD	104	94	2^{30}	2^{-15}	518.4 KB	17.2 KB	124.3 ms	74.64 ms	4.772 hour
Scalable Fuzzy Tracking	115	104	2^{20}	2^{-10}	800 B	800 B	0.011 ms	0.0148 ms	3.424 ms
Scalable Fuzzy Tracking	115	104	2^{30}	2^{-15}	800 B	800 B	0.011 ms	0.0149 ms	108.77 ms

[1] Only consider the fuzzy part, i.e., the time to generate the fuzzy tracking information, ftki.
[2] The server's running time for each incoming ftki.
For Post-quantum FMD, we calculate the time to run FTrack for all of clients (recipients).

FMD in Section 5.5.2, to get 104-bit computational security and 40-bit statistical security, we choose $q = 2^{60}, \ell = 2304$ and $\chi = B_\eta$ is binomial distribution with parameter $\eta = 3$. For Scalable Fuzzy Tracking in Section 5.5.3, to get 115-bit security and negligible failure probability, we choose $q = 4096$, other parameters are the same as Kyber512, specifically, we have $m = 256, \eta = 3, \ell = 2$.

We run the implementation on a regular laptop: Macbook Air (M1 2020) with 8GB RAM and 2.1 GHz CPU (Turbo 3.2 Ghz). Note that our implementation is based on the reference implementation of Dilithium, Kyber, and Falcon, without using AES or AVX optimization. We run each test 10000 times to calculate its average running time. For Post-quantum FMD, we run tests 100 times to average the running time.

Experimental results show that Falcon512+Spirit2 yields the smallest signature size (4.09 KB) for security against key exposures with a decent hardness level (114-bit security). And Scalable Fuzzy Tracking yields the smallest communicational cost (800 Bytes) and server's computational overhead (3.4 ms) for millions of clients.

5.6.2 Prior Works

We also present tables for comparison with prior works in Table 5.1 and Table 5.2.

In Table 5.1 we compare our group-based stealth signature (Section A.2), Spirit2 (Section 5.5.1), Spirit2 +Dilithium2, and Spirit2 +Falcon512 with prior works. We would like to stress that [LLN+20] is a theoretical work without giving concrete parameters. We estimate the number as follows: According to Lemma 5 in Section 2.2, it requires $m \geq 6n \log q$, and according to Section 3.4, it requires $q = \tilde{O}(m^{5/2}) \cdot \mathsf{superpoly}(\log m)$. Concretely, if we choose the security parameter $n = 2^{10}$ which is the case in our thesis, a typical choice for [LLN+20] to satisfy all of the above conditions is $n = 2^{10}, m = 2^{18}, q = 2^{50}$.

If we want to improve their work with recent advancement in NTRU, note that the techniques used in [LLN+20] is from [ABB10] which implies HIBE. Though combining it with NTRU could improve its efficiency, however, it is highly likely to have similar parameters as the state of the art about NTRU-based HIBE [ZMS+21]. Thus we estimate numbers here with parameters from [ZMS+21] for 80-bit security since they only have two levels of security (80-bit or 160-bit).

In Table 5.2 we compare our Post-quantum FMD (Section 5.5.2) and Scalable Fuzzy Tracking (Section 5.5.3) with prior works about message detection or retrieval. All of the works assume the semi-honest server, except that Π_{TEE} also considers the malicious server. Note that for security, ρ needs to be as large as $\frac{1}{\sqrt{N}}$ as calculated in [SPB21]. Also, some prior works consider fuzzy schemes ([BLMG21] and ours) are ρM-anonymity where M is the number of total messages. However, it is not accurate due to statistical attacks as shown in [SPB21]: If there is only one message ($M = 1$), it still has some extent of anonymity if N is large.

Regarding the server's work, we compare a single server with a single thread as all of the works (except for Π_{GC}) supporting distributed servers or parallelized threads. [BLMG21] actually needs to run their test functionality for each recipient's detection key for each incoming message. Other schemes with full privacy inherently require $O(N)$ work for the server otherwise it will leak information.

Latency per message is dominated by the server's computational time. Assuming there are $N = 2^{20}$ clients (millions of users is a legit assumption for cryptocurrency [Webf]). Set false-positive rate $\rho = 2^{-10}$ for [BLMG21] and ours. The numbers of others are taken from their paper directly. Assuming $10 - 20$ messages per second (e.g., Bitcoin or Ethereum), only ours is practical with many users. To compute each recipient's computational time, we assume there are $M = 500,000$ messages in total (which is roughly the number of transactions of Bitcoin or Ethereum per day). We let fuzzy tracking schemes run Track (shown in Definition 5.3.1) for each message retrieved.

5.7 Security Analysis

Proof of Theorem 5.4.1 We restate the theorem here more formally for the case of unforgeability.

Theorem 5.7.1. *The stealth signature* $\mathsf{SS_w}$ *constructed in Section 5.4 is secure in sEUF-CMA*$_{\mathsf{w-ke}}$ *experiment if* $\mathsf{SS_{w/o}}$ *is EUF-CMA*$_{\mathsf{w/o-ke}}$ *secure and* DS *is sEUF-CMA secure. Specifically, for any* $\lambda \in \mathbb{N}$, *and for any PPT adversary* $\mathcal{A}$, *if it succeeds in the experiment sEUF-CMA*$_{\mathsf{w-ke}}$, *then there are other adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *running in roughly same time such that*

$$\mathsf{Adv}_{\lambda}^{sEUF\text{-}CMA_{\mathsf{w-ke}}}(\mathcal{A}) \le \mathsf{Adv}_{\lambda}^{EUF\text{-}CMA_{\mathsf{w/o-ke}}}(\mathcal{B}_1) + \mathsf{Adv}_{\lambda}^{sEUF\text{-}CMA}(\mathcal{B}_2).$$

Proof. We prove the theorem by reduction. Suppose there's an adversary $\mathcal{A}$ has a non-negligible advantage in sEUF-CMA$_{\mathsf{w-ke}}$, then we can construct another adversary $\mathcal{B}$ to win the experiment EUF-CMA$_{\mathsf{w/o-ke}}$ of $\mathsf{SS_{w/o}}$ or the experiment sEUF-CMA (strong unforgeability) of DS as follows. $\mathcal{B}$ forwards $\mathsf{mpk}, \mathsf{mtk}$ from the challenger in EUF-CMA$_{\mathsf{w/o-ke}}$ to $\mathcal{A}$.

To simulate $\mathsf{OSKGen}\mathcal{O}(i, \mathsf{opk}^i, \mathsf{tki}^i, \mathsf{flag}^i)$, if $\mathsf{flag}^i = \mathsf{true}$, $\mathcal{B}$ runs $(\mathsf{vk}^i, \mathsf{sk}^i) \leftarrow \mathsf{DS.Gen}$, then queries $\sigma_1^i \leftarrow \mathsf{Sign}\mathcal{O}(\mathsf{vk}^i)$ in EUF-CMA$_{\mathsf{w/o-ke}}$ and returns $\mathsf{osk}^i := (\sigma_1^i, \mathsf{vk}^i, \mathsf{sk}^i)$ to $\mathcal{A}$; If $\mathsf{flag}^i = \mathsf{false}$, $\mathcal{B}$ asks a challenger $\mathcal{C}^i$ in sEUF-CMA of DS to send a challenge verification key vk^i, then queries $\sigma_1^i \leftarrow \mathsf{Sign}\mathcal{O}(\mathsf{vk}^i)$ in sEUF-CMA$_{\mathsf{w/o-ke}}$ of $\mathsf{SS_{w/o}}$ and stores σ_1^i; If $\mathsf{OK}[i] = (\mathsf{opk}^i, \cdot, \cdot) \wedge \mathsf{flag}^i = \mathsf{true}$, $\mathcal{B}$ signals $\mathcal{C}^i$ to terminal the experiment and asks for its osk^i then forwards that to $\mathcal{A}$. To simulate $\mathsf{Sign}\mathcal{O}(i, m^j)$, $\mathcal{B}$ queries $\sigma_2^j \leftarrow \mathsf{Sign}\mathcal{O}(m^j || \sigma_1^i)$ and returns $\sigma^j := (\sigma_1^i, \sigma_2^j, \mathsf{vk}^i)$ to $\mathcal{A}$.

Once $\mathcal{A}$ submits some valid forgery $\sigma' := (\sigma_1', \sigma_2', \mathsf{vk}'), m', i'$ as shown in Figure 5.3, $\mathcal{B}$ behaves in following cases:

- If m' is not appeared in Q (recall that Q is the set to record signing queries), $\mathcal{B}$ forwards $\sigma_2', m'||\sigma_1'$ to i'-th challenger in sEUF-CMA of DS;
- If vk' is not appeared in Q, $\mathcal{B}$ forwards σ_1', vk' to the challenger in EUF-CMA$_{\mathsf{w/o-ke}}$ of $\mathsf{SS}_{\mathsf{w/o}}$;
- If both m', vk' are in Q, then the only case that σ' is a valid forgery is either σ_1' or σ_2' not appeared in Q. In either case, $\mathcal{B}$ just forwards $\sigma_2', m'||\sigma_1'$ to the challenger in sEUF-CMA of DS.

This completes the proof.

$\square$

We restate the theorem here for unlinkability.

Theorem 5.7.2. *The stealth signature SS_{w} constructed in Section 5.4 is secure in UNLNK$_{\mathsf{w-ke}}$ experiment if $\mathsf{SS}_{\mathsf{w/o}}$ is UNLNK$_{\mathsf{w/o-ke}}$ secure. Specifically, for any $\lambda \in \mathbb{N}$, and for any PPT adversary $\mathcal{A}$, if it succeeds in the experiment UNLNK$_{\mathsf{w-ke}}$, then there are other adversaries $\mathcal{B}$ running in roughly same time such that*

$$\mathsf{Adv}_{\lambda}^{UNLNK_{\mathsf{w-ke}}}(\mathcal{A}) \leq \mathsf{Adv}_{\lambda}^{UNLNK_{\mathsf{w/o-ke}}}(\mathcal{B}).$$

Proof. Similarly, we can also prove this theorem easily by reduction. Suppose there's an adversary $\mathcal{A}$ has a non-negligible advantage in UNLNK$_{\mathsf{w-ke}}$, then we can construct another adversary $\mathcal{B}$ to win the experiment UNLNK$_{\mathsf{w/o-ke}}$ of $\mathsf{SS}_{\mathsf{w/o}}$ as follows. $\mathcal{B}$ forwards $\mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{opk}_b, \mathsf{tki}_b$ from the challenger in UNLNK$_{\mathsf{w/o-ke}}$ to $\mathcal{A}$. To simulate osk_b, $\mathcal{B}$ runs $\mathsf{DS.Gen}$ to get $(\mathsf{vk}, \mathsf{sk})$, then queries the signing oracle via $\mathsf{Sign}\mathcal{O}(\cdot, -1, \mathsf{vk})$ from UNLNK$_{\mathsf{w/o-ke}}$ to learn a signature σ_1 of vk, then returns $\mathsf{osk}_b := (\sigma_1, \mathsf{vk}, \mathsf{sk})$ to $\mathcal{A}$. To simulate $\mathsf{OSKGen}\mathcal{O}$, $\mathcal{B}$ queries $\mathsf{Sign}\mathcal{O}$ and runs $\mathsf{DS.Gen}$ as above to generate osk. Once $\mathcal{A}$ submits b', $\mathcal{B}$ simply forwards b' as its final guess. This completes the proof.

$\square$

5.7.1 Security Analysis of Stealth Signature Without Fuzzy Tracking

Proof of Lemma 5.5.1 We restate the lemma formally here.

Lemma 5.7.3. *SPIRIT in Figure 5.8 is unforgeable without key exposure under no-message attacks. Specifically, in random oracle model, for any $\lambda \in \mathbb{N}$, for any adversary $\mathcal{A}$, if Dil has parameters $\beta, \gamma_1, \gamma_2$, and we denote H' as a random oracle can be accessed by $\mathcal{A}$ and $\mathcal{B}_2$, then the advantage to win the game UF-NMA$^{\mathcal{A}}_{\mathsf{w/o-ke}}(\lambda)$ is*

$$\mathsf{Adv}_{\lambda,\mathsf{H}',\gamma_1,\gamma_2,\beta}^{UF\text{-}NMA_{\mathsf{w/o-ke}}}(\mathcal{A}) \leq \mathsf{Adv}_{k,\ell,D}^{\mathsf{MLWE}}(\mathcal{B}_1) \qquad + \mathsf{Adv}_{\mathsf{H}',k,\ell+1,\zeta}^{\mathsf{SelfTargetMSIS}}(\mathcal{B}_2).$$

Proof. Consider the experiment EUF-CMA$_{\mathsf{w/o-ke}}$ in Figure 5.1 where the $\mathsf{Sign}\mathcal{O}$ is forbidden to access. Suppose $\mathcal{A}$ forges $\sigma*$, then we have the following claim.

Claim 5.7.1. *If an adversary $\mathcal{A}$ can forge σ^* without accessing* $\mathsf{Sign}\mathcal{O}$ *and assuming* $\mathsf{MLWE}_{k,\ell,D}$ *assumption holds, then there is another adversary $\mathcal{B}_2$ who solves* $\mathsf{SelfTargetMSIS}_{\mathsf{H}',k,\ell+1,\zeta}$ *in roughly same time with non-negligible probability.*

Proof. After receiving uniformly random samples $(\mathbf{A}, \mathbf{t}) \in R^{k\times\ell} \times R^k$ and random oracle access $\mathsf{H}'(\cdot)$ from the challenger in $\mathsf{SelfTargetMSIS}_{\mathsf{H}',k,\ell+1,\zeta'+\beta}$, $\mathcal{B}_2$ computes $\mathsf{mpk} := (\mathbf{A}, \mathbf{t}, \mathsf{ek})$, $\mathsf{mtk} := (\mathsf{dk}, \mathbf{t})$ and forwards $\mathsf{mpk}, \mathsf{mtk}, \mathsf{H}'$ to $\mathcal{A}$. As long as the $\mathsf{MLWE}_{k,\ell,D}$ assumption holds, mpk looks indistinguishable from real public key for $\mathcal{A}$. For i-th query in $\mathsf{OSKGen}\mathcal{O}$, $\mathcal{B}_2$ computes and stores $\mathbf{s}_1^i, \mathbf{s}_2^i$. Once $\mathcal{A}$ submits some valid forgery σ^* with i^*, meaning it finds some $(\mathbf{x}, \mathbf{z}, c)$ for $\mathsf{opk}^* := \mathbf{t}^*$ such that

$$\mathsf{H}'\left(\mu \parallel [\mathbf{I}_k|\mathbf{A}|\mathbf{t}^*]\cdot\begin{bmatrix}\mathbf{x}\\ \mathbf{z}\\ c\end{bmatrix}\right) = c,$$

where $\|\mathbf{x}\|_\infty \le 2\gamma_2 + 1 + 2^{d-1}\tau$, $\|\mathbf{z}\|_\infty \le \gamma_1 - 2\beta$ and $\|c\|_\infty = 1$[LDK+20]. Then $\mathcal{B}_2$ can retrieve $\mathbf{s}_1^*, \mathbf{s}_2^*$ from its storage and instantly return $\mathbf{y} := \begin{bmatrix}\mathbf{x}'\\ \mathbf{z}'\\ c\end{bmatrix}$, μ to the $\mathsf{SelfTargetMSIS}_{\mathsf{H}',k,\ell+1,\zeta}$ challenger, where $\mathbf{x}' := \mathbf{x} + c\mathbf{s}_2^*$ and $\mathbf{z}' := \mathbf{z} + c\mathbf{s}_1^*$. Note that $\|c\mathbf{s}_1^*\|, \|c\mathbf{s}_2^*\| \le \beta$. Since we can write $\mathbf{t}^* := \mathbf{t} + \mathbf{A}\mathbf{s}_1^* + \mathbf{s}_2^*$, it is easy to check that this is a valid solution

$$\mathsf{H}'\left(\mu \parallel [\mathbf{I}_k|\mathbf{A}|\mathbf{t}]\cdot\begin{bmatrix}\mathbf{x} + c\mathbf{s}_2^*\\ \mathbf{z} + c\mathbf{s}_1^*\\ c\end{bmatrix}\right) = c$$

where $\|\mathbf{y}\|_\infty \le \zeta$ and $\zeta := \max\{\gamma_1 - \beta, 2\gamma_2 + 1 + 2^{d-1}\tau + \beta\}$. □

This completes the proof to show it is secure in the UF-NMA$_{\mathsf{w/o-ke}}$ experiment. □

Proof of Theorem 5.5.2 We restate the theorem for unforgeable without key exposures formally here.

Theorem 5.7.4. *Spirit in Figure 5.8 is existential unforgeable without key exposures. Specifically, for any adversary $\mathcal{A}$, if it succeeds in the experiment EUF-CMA$_{\mathsf{w/o-ke}}$, then there is another adversary $\mathcal{B}$ running in roughly same time such that*

$$\mathsf{Adv}_{\lambda,\mathsf{H},\gamma_1,\gamma_2,\beta}^{\textit{EUF-CMA}_{\mathsf{w/o-ke}}}(\mathcal{A}) \le \mathsf{Adv}_{\lambda,\mathsf{H}',\gamma_1,\gamma_2,\beta}^{\textit{UF-NMA}_{\mathsf{w/o-ke}}}(\mathcal{B}) + \mathsf{negl}(\lambda),$$

where we denote H', H *as random oracles can be accessed by $\mathcal{B}_1$ and $\mathcal{A}$, respectively.*

Proof. Intuitively, a reduction from CMA to NMA usually needs "patching" random oracles [KLS18, AFLT12]. We prove this theorem in a sequence of hybrid games as follows.
Hybrid$_0$: This is exactly the standard EUF-CMA$_{\mathsf{w/o-ke}}$ experiment. Thus we have

$$\Pr[\mathbf{Hybrid}_0 \Rightarrow 1] = \mathsf{Adv}_{\lambda,\mathsf{H},\gamma_1,\gamma_2,\beta}^{\text{EUF-CMA}_{\mathsf{w/o-ke}}}(\mathcal{A}).$$

```
$\mathsf{H}(\mathbf{w}_1||\mu)$
    / in $Hyb_2$ and $Hyb_3$
Retrieve $\langle \mu : (c^\mu, \mathbf{w}_1^\mu \rangle$ for $\mu$
if $\mathbf{w}_1 = \mathbf{w}_1^\mu$
    then return $c := c^\mu$
else return $c := \mathsf{H}'(\mathbf{w}_1||\mu)$
```

```
$\text{EUF-CMA}^{\mathcal{A}}_{\mathsf{w/o-ke}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mtk}) \leftarrow \mathsf{MKGen}(\lambda)$
$\mathsf{OK} := [], Q := \emptyset$
$(m^*, \sigma^*, i^*) \leftarrow \mathcal{A}^{\mathsf{OSKGen}\mathcal{O}, \mathsf{Sign}\mathcal{O}}(\mathsf{mpk}, \mathsf{mtk})$
$(\mathsf{opk}^* := \mathbf{t}^*, \mathsf{osk}^*, \cdot) := \mathsf{OK}[i^*]$
    / $Hyb_3$ block begins
$(\mathbf{z}^*, c^*, \mathbf{h}^*) := \sigma^*$
$\mu^* \leftarrow \mathsf{G}(m^*||\mathbf{t}^*)$
$\mathbf{w}_1^* \leftarrow \mathsf{HighBits}_q(\mathbf{Az}^* - c^*\mathbf{t}^*, 2\gamma_2)$
if $\mathsf{H}'(\mathbf{w}_1^*||\mu^*) \neq c^*$
    then return 0
    / $Hyb_3$ block ends
$b_0 := (m^*, i^*) \notin Q$
$b_1 := \mathsf{Vf}(\mathsf{opk}^*, m^*, \sigma^*) = 1$
$b_2 := (\mathsf{OK}[i^*] \neq (\cdot, \cdot, \perp))$
return $b_0 \wedge b_1 \wedge b_2$
```

Figure 5.11: Simulated H and $\text{EUF-CMA}^{\mathcal{A}}_{\mathsf{w/o-ke}}(\lambda)$ in $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$

$\mathbf{Hybrid}_1$**:** We modify $\mathbf{Hybrid}_0$ as follows. In $\mathsf{OSKGen}\mathcal{O}(\mathsf{opk}^i, \mathsf{tki}^i)$, for i-th query, if $\mathsf{true} \leftarrow \mathsf{Track}(\mathsf{mtk}, \mathsf{opk}^i, \mathsf{tki}^i)$ it only stores $\mathbf{s}_1^i, \mathbf{s}_2^i, \mathbf{t}^i := \mathbf{A}\mathbf{s}_1^i + \mathbf{s}_2^i + \mathbf{t}$, sets $\mathsf{osk}^i := \top$ and returns 1. In $\mathsf{Sign}\mathcal{O}(i, m^j)$, for j-th query, it generates and sets osk^i by msk if $\mathsf{osk}^i := \top$, then return a signature σ^j by using osk^i.

This game only changes the time to generate osk^i, thus advantage remains the same:

$$|\Pr[\mathbf{Hybrid}_1 \Rightarrow 1] - \Pr[\mathbf{Hybrid}_0 \Rightarrow 1]| = 0.$$

$\mathbf{Hybrid}_2$**:** We update $\mathbf{Hybrid}_1$ by modifying $\mathsf{Sign}\mathcal{O}(i, m^j)$ in j-th query: Instead of generating σ^j with osk^i when needed, it just simulates σ^j by choosing uniformly random $(\mathbf{z}^j, c^j) \in S^{\ell}_{\gamma_1 - 2\beta - 1} \times B_\tau$ and stores a key-value pair $\langle \mu^j : (c^j, \mathbf{w}_1^j) \rangle$ where $\mu^j \leftarrow \mathsf{G}(m^j||\mathbf{t}^i), \mathbf{w}_1^{\mu^j} \leftarrow \mathsf{HighBits}_q(\mathbf{Az}^j - c^j\mathbf{t}^i, 2\gamma_2)$, and G is a perfect random function. We also use a new random oracle $\mathsf{H}(\mathbf{w}_1||\mu)$ to simulate random oracle $\mathsf{H}'(\mathbf{w}_1||\mu)$ in above game as shown in left part of Figure 5.11. Now we analyze the advantage. In our construction, $\mathsf{Dil.Sign}$ remains unaltered, thus the resulting signature σ is still perfectly zero-knowledge (where the exact simulation is shown in Sign of Figure 5.12). Therefore the distribution of each σ is exactly the same as the one in $\mathbf{Hybrid}_1$, then we have

$$|\Pr[\mathbf{Hybrid}_2 \Rightarrow 1] - \Pr[\mathbf{Hybrid}_1 \Rightarrow 1]| = 0.$$

$\mathbf{Hybrid}_3$**:** We modify the above game by adding an additional block in $\text{EUF-CMA}_{\mathsf{w/o-ke}}$ as shown in the right part of Figure 5.11. This game only differs from the $\mathbf{Hybrid}_2$ if $\mathbf{w}_1^* = \mathbf{w}_1^{\mu^*}$ and $((m^*, \cdot, i^*) \notin Q) \wedge$

$b_1 \wedge b_2$ ($\mathbf{Hybrid}_3$ return 0 and $\mathbf{Hybrid}_2$ return 1). However, $\mathcal{A}$ didn't query $\mathsf{Sign}\mathcal{O}(i^*, m^*)$ before, thus $\mathbf{w}_1^{\mu^*}$ should remain hidden. And from [KLS18], it shows that Dil signature has enough min-entropy, thus the probability $\Pr[\mathbf{w}_1^* = \mathbf{w}_1^{\mu^*}]$ is negligible, i.e.,

$$|\Pr[\mathbf{Hybrid}_3 \Rightarrow 1] - \Pr[\mathbf{Hybrid}_2 \Rightarrow 1]| \leq \mathsf{negl}(\lambda).$$

This game can be fully simulated by $\mathcal{B}$ against UF-NMA$_{\mathsf{w/o-ke}}$ as follows. $\mathcal{B}_1$ simulates $\mathsf{OSKGen}\mathcal{O}$, $\mathsf{Sign}\mathcal{O}$ oracles without knowing msk, and it patches H' from UF-NMA$_{\mathsf{w/o-ke}}$ to H for generating σ^i. Once $\mathcal{A}$ submits a valid signature σ^* and if H' works well in σ^*, $\mathcal{B}$ directly forwards σ^* to the challenger of UF-NMA$_{\mathsf{w/o-ke}}$. Therefore

$$\Pr[\mathbf{Hybrid}_3 \Rightarrow 1] = \mathsf{Adv}_{\lambda,\mathsf{H}',\gamma_1,\gamma_2,\beta}^{\text{UF-NMA}_{\mathsf{w/o-ke}}}(\mathcal{B}_1)$$

and we complete the proof. □

We state the theorem for unlinkable without key exposures formally here.

Theorem 5.7.5. *SPIRIT in Figure 5.8 is unlinkable without key exposures. Specifically, for any adversary $\mathcal{A}$, if it succeeds in the experiment UNLNK$_{\mathsf{w/o-ke}}$, then there are other adversaries $\mathcal{B}_1, \mathcal{B}_2$ running in roughly same time such that*

$$\mathsf{Adv}_{\lambda,\mathsf{H},\gamma_1,\gamma_2,\beta}^{\mathit{UNLNK}_{\mathsf{w/o-ke}}}(\mathcal{A}) \leq \mathsf{Adv}_{\lambda}^{\mathit{ANO\text{-}CCA}}(\mathcal{B}_1) + \mathsf{Adv}_{k,\ell,D}^{\mathsf{MLWE}}(\mathcal{B}_2).$$

where we denote H *as a random oracles can be accessed by $\mathcal{A}$ and $\gamma_1, \gamma_2, \beta$ are parameters of the underlying* Dil *scheme.*

Proof. We prove the theorem in a sequence of hybrid games.
$\mathbf{Hybrid}_0$**:** This is the original UNLNK$_{\mathsf{w-ke}}$ experiment, thus we have

$$\Pr[\mathbf{Hybrid}_0 \Rightarrow 1] = \mathsf{Adv}_{\lambda,\mathsf{H},\gamma_1,\gamma_2,\beta}^{\text{UNLNK}_{\mathsf{w/o-ke}}}(\mathcal{A}).$$

$\mathbf{Hybrid}_1$**:** We modify the above game by changing the function $\mathsf{Sign}(\mathsf{osk}, m^j)$ in UNLNK$_{\mathsf{w/o-ke}}$ experiment to the $\mathsf{Sign}(\mathsf{opk}^i, m^j)$ without using osk in Figure 5.12. Specifically, it samples uniformly random $(\mathbf{z}^j, c^j)$, programs the random oracle such that $\mathsf{H}(\mu^j || \mathbf{w}_1^j) = c^j$ where μ^j is determined by m^j and $\mathbf{w}_1^j := \mathsf{HighBits}(\mathbf{A}\mathbf{z}^j - c^j\mathbf{t}^i, 2\gamma_2)$. Then set $\sigma^j := (\mathbf{z}^j, c^j, \mathbf{h}^j)$ where $\mathbf{h}^j$ can be determined by $c^j, \mathbf{t}^i, \mathbf{z}^j$. Because of the perfectly zero-knowledge of σ^j, the distribution of signatures in this hybrid is the same as the one in $\mathbf{Hybrid}_0$, i.e.,

$$\left|\Pr[\mathbf{Hybrid}_0 \Rightarrow 1] - \Pr[\mathbf{Hybrid}_1 \Rightarrow 1]\right| = 0.$$

$\mathbf{Hybrid}_2$**:** We modify the above game as follows. Parse $\mathsf{mpk}_0 := (\mathbf{t}_0, \mathsf{ek}_{1,0})$ and $\mathsf{mpk}_1 := (\mathbf{t}_1, \mathsf{ek}_{1,1})$, instead of generating $\mathbf{t}_0, \mathbf{t}_1$ from msk, we sample uniformly random $(\mathbf{t}_0, \mathbf{t}_1) \leftarrow\!\!\$\ R_q^k \times R_q^k$. By $\mathsf{MLWE}_{k,\ell,D}$

```
Sign(opk^i, m^j)
    // in $Hyb_1$ and $Hyb_2$
parse $\mathbf{t}^i := \mathsf{opk}^i$
$(\mathbf{z}^i, c^i) \leftarrow\!\$\ S^{\ell}_{\gamma_1 - 2\beta_1} \times B_\tau$
$\mu^i \leftarrow \mathsf{G}(m^j || \mathbf{t}^i)$
$\mathbf{w}^i_1 \leftarrow \mathsf{HighBits}_q(\mathbf{A}\mathbf{z}^i - c^i\mathbf{t}^i, 2\gamma_2)$
Program H s.t. $\mathsf{H}(\mathbf{w}^i_1 || \mu^i_1) := c^i$
$\mathbf{h}^i \leftarrow \mathsf{MakeHint}_q(-c^i\mathbf{t}^i_0, \mathbf{A}\mathbf{z}^i - c^i\mathbf{t}^i + c^i\mathbf{t}^i_0, 2\gamma_2)$
    // $\mathbf{t}^i_0$ are lower bits of $\mathbf{t}^i$
return $\sigma^i := (\mathbf{z}^i, c^i, \mathbf{h}^i)$
```

Figure 5.12: Simulation of Sign from $\mathbf{Hybrid}_1$ to $\mathbf{Hybrid}_2$

assumption, we know this hybrid only differs from $\mathbf{Hybrid}_1$ by:

$$\left|\Pr[\mathbf{Hybrid}_2 \Rightarrow 1] - \Pr[\mathbf{Hybrid}_1 \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{MLWE}}_{k,\ell,D}(\mathcal{B}_2).$$

Besides, this hybrid can be fully simulated by an adversary $\mathcal{B}_1$ of ANO-CCA experiment. $\mathcal{B}_1$ simulates the random oracle H for $\mathcal{A}$. Upon receiving $\mathsf{ek}_0, \mathsf{ek}_1$ and (C_b, K_b) from ANO-CCA experiment, $\mathcal{B}_1$ sets $\mathsf{mpk}_0 := (\mathbf{t}_0, \mathsf{ek}_0)$ and $\mathsf{mpk}_1 := (\mathbf{t}_0, \mathsf{ek}_1)$ where $(\mathbf{t}_0, \mathbf{t}_1) \leftarrow\!\$\ R^k_q \times R^k_q$ are uniformly sampled. $\mathcal{B}_1$ sets $\mathsf{tki}_b := C_b, \mathsf{osk}_b := \top, \mathsf{opk}_b \leftarrow\!\$\ R^k_q$, and sends $(\mathsf{mpk}_0, \mathsf{mpk}_1, \mathsf{tki}_b, \mathsf{opk}_b)$ to $\mathcal{A}$ of $\mathbf{Hybrid}_2$. For each query of $\mathsf{OSKGen}\mathcal{O}(b^*, \mathsf{opk}^i, \mathsf{tki}^i)$, $\mathcal{B}_1$ queries $K^i \leftarrow \mathsf{KEM.Decaps}\mathcal{O}(b^*, \mathsf{tki}^i)$ to check if $\mathbf{A}\mathbf{s}^i_1 + \mathbf{s}^i_2 + \mathbf{t}_{b^*} = \mathsf{opk}^i$ where $\mathbf{s}^i_1, \mathbf{s}^i_2 \leftarrow \mathsf{Dil.ExpandS}(K^i)$. If the check doesn't pass, set $\mathsf{osk}^i_{b^*} :=\perp$; Otherwise set $\mathsf{osk}^i_{b^*} := \top$. For each query of $\mathsf{Sign}\mathcal{O}(b^*, i, m^j)$, $\mathcal{B}_1$ simulates the signature σ^j by using $\mathsf{opk}^i_{b^*}$ if the corresponding $\mathsf{osk}^i_{b^*} = \top$, otherwise return $\perp$. If $i = -1$, just simulates a signature using opk_b.

Then $\mathbf{Hybrid}_2$ can be simulated without knowing any $\mathsf{msk}, \mathsf{mtk}$ or b. Once $\mathcal{A}$ returns b', $\mathcal{B}_1$ simply forwards b' to the challenger of ANO-CCA. Thus we have

$$\Pr[\mathbf{Hybrid}_2 \Rightarrow 1] = \mathsf{Adv}^{\text{ANO-CCA}}_{\lambda}(\mathcal{B}_1),$$

and this completes the proof. □

5.7.2 Analysis of Post-quantum FMD

Correctness. We show the scheme in Figure 5.9 satisfies Definition 5.3.8 as follows. For each $i \in [t]$, we have $\lceil \mathbf{s}^T_i \mathbf{c}_1 + z \rfloor_2 \oplus w_i = 1$. Since $c_{2,i} - \mathbf{s}^T_i \mathbf{c}_1 = \frac{q}{2} + e'$ where $e' \in [-B, B]$ is some short error,

```
pRgv.Gen(λ, n)
    $\mathbf{A} \in \mathbb{Z}_q^{\ell \times \ell} \leftarrow\$ \mathsf{crs}$
    $(\mathbf{S}, \mathbf{E}) \leftarrow\$ (\chi^{\ell \times n})^2$
    $\mathbf{B} := \mathbf{AS} + \mathbf{E}$
    return $\mathsf{pk} := \mathbf{B}, \mathsf{sk} := \mathbf{S}$

pRgv.Dec(sk, ct)
    parse $(\mathbf{c}_1, z, w_1, \dots, w_n) := \mathsf{ct}$
    parse $(\mathbf{s}_1, \dots, \mathbf{s}_t) := \mathsf{sk}$
    $\forall i \in [t], m_i := \lceil \mathbf{s}_i^T \mathbf{c}_1 + z \rfloor_2 \oplus w_i$
    return $\mathbf{m} := [m_1, \dots, m_t]$

pRgv.Enc(pk, $\mathbf{m} \in \{0,1\}^n$)
    $(\mathbf{r}, \mathbf{e}_1) \leftarrow\$ (\chi^{\ell})^2, \mathbf{e}_2 \leftarrow\$ \chi^n$
    $\mathbf{c}_1 := \mathbf{A}^T \mathbf{r} + \mathbf{e}_1$
    $\mathbf{c}_2 := \mathbf{B}^T \mathbf{r} + \mathbf{e}_2 + \frac{q}{2} \cdot \mathbf{m}$
    $z \leftarrow\$ \mathbb{Z}_q$ such that $\forall i \in [n]$ :
        $z + c_{2,i} \notin [\frac{q}{4} - B, \frac{q}{4} + B] \cup$
            $[\frac{3q}{4} - B, \frac{3q}{4} + B]$
    $\forall i \in [n], w_i := \lceil z + c_{2,i} \rfloor_2$
    return $\mathsf{ct} := (\mathbf{c}_1, z, w_1, \dots, w_n)$
```

Figure 5.13: Packed Regev encryption pRgv with ciphertext compression

we have $c_{2,i} - e' = \mathbf{s}_i^T \mathbf{c}_1 + \frac{q}{2}$. Also, we choose

$$c_{2,i} + z \notin [\frac{q}{4} - B, \frac{q}{4} + B] \cup [\frac{3q}{4} - B, \frac{3q}{4} + B],$$

thus we have $\lceil c_{2,i} + z \rfloor_2 = \lceil c_{2,i} + z - e' \rfloor_2$, which implies $w_i = \lceil \mathbf{s}_i^T \mathbf{c}_1 + z + \frac{q}{2} \rfloor_2 = \lceil \mathbf{s}_i^T \mathbf{c}_1 + z \rfloor_2 \oplus 1$. Therefore, with correct ftk, FTrack always returns true. Note that for correctness, we require $q > 4Bn$.

Security Analysis. We show the scheme in Figure 5.9 is unlinkable with key-exposure and fuzzy tracking (Definition 5.3.9).

Proof of Lemma 5.5.3 and Theorem 5.5.4 We restate the formal lemma here.

Lemma 5.7.6. *Packed Regev encryption* pRgv *with ciphertext compression shown in Figure 5.13 satisfies Definition 1.1.6 and is uniformly-ambiguous UNI-AMB-secure when* $\frac{4Bn}{q}$ *is* $\mathsf{negl}(\lambda)$. *Specifically, we have*

$$\mathsf{Adv}_{\lambda}^{\mathit{UNI\text{-}AMB}}(\mathcal{A}) \leq \mathsf{Adv}_{\ell,q}^{\mathsf{LWE}}(\mathcal{A}) + \frac{4Bn}{q},$$

where B *is the bound such that* $\left\|\mathbf{S}^T \mathbf{c}_1 - \mathbf{c}_2\right\|_\infty \mod \frac{q}{2} < B$.

Proof. To see it is uniformly ambiguous, firstly note that $\mathbf{c}_1$ looks uniformly random due to LWE assumption, and w_i is uniformly random due to m_i being a uniformly random bit in the experiment in Figure 1.2. For z, the statistical distance between its distribution and uniformly random distribution over $\mathbb{Z}_q$ is $\frac{4Bn}{q}$. Thus as long as $\frac{4Bn}{q} \leq \mathsf{negl}(\lambda)$, we can simulate the entire ciphertext without knowing b or sk. □

We restate the theorem formally here.

Theorem 5.7.7. *The fuzzy stealth signature constructed in Figure 5.9 is* unlinkable with key exposure and fuzzy tracking. *Specifically, for any λ, n, t where $n \geq t$, if there is a PPT adversary $\mathcal{A}$ has non-negligible advantage in experiment defined in Figure 5.5, then there exist other adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ running in roughly same time such that:*

$$\mathsf{Adv}_{\lambda,n,t}^{UNLNK_{\mathsf{fw-ke}}}(\mathcal{A}) \leq 2\mathsf{Adv}_{\lambda}^{UNLNK_{\mathsf{w-ke}}}(\mathcal{B}_1) + \\ p(\lambda) \cdot \left(4t\mathsf{Adv}_{\lambda}^{UNI\text{-}AMB}(\mathcal{B}_2) + (n-t)\mathsf{Adv}_{\lambda}^{IK\text{-}CPA}(\mathcal{B}_3)\right),$$

where $p(\lambda)$ is some polynomial on security parameter λ.

Proof. Combined with Lemma 5.7.6, recall Theorem 11 and Lemma 2 in [BLMG21] to prove this via the same approach. □

Extends to finer false-positive rates. We introduce an approach to achieve finer false-positive rates ($\rho \neq \frac{1}{2^t}$) in fuzzy tracking (and also FMD) schemes. As mentioned in [BLMG21], to achieve finer rates like $\frac{1}{3}, \frac{1}{5}$ is easy via switching the base. However, achieving rates like $\frac{3}{4}$ is still challenging without garbled circuits. We show how to achieve a rate like $\frac{\alpha}{2^k}$ where $1 \leq \alpha \leq 2^k - 1$ with a small tweak but α, k needs to be fixed in advance. The sender instead of computing $\mathsf{Enc}(\mathsf{pk}_i, 1)$ for each $i \in [n]$, it computes $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{msg}_i)$ where msg_i is uniformly sampled via $\mathsf{msg}_i \leftarrow\$ \{0, 1, \ldots, \alpha\}$. The detector only accepts the ciphertext c_i if and only if $\mathsf{Dec}(\mathsf{sk}_i, c_i) \leq \alpha$. It is easy to see that this satisfies correctness, fuzziness and security simultaneously and is compatible with $\mathsf{FMD}_1, \mathsf{FMD}_2$ in [BLMG21] and our fuzzy tracking scheme in Figure 5.9. Essentially, the receiver is able to 'tune' the false-positive rate ρ via a finer step: Originally, ρ can only be decreased half by half (i.e., from ρ to $\frac{\rho}{2}$ each time); Now it can be decreased by a factor $\frac{\alpha}{2^k}$ (i.e., from ρ to $\frac{\rho\alpha}{2^k}$). For example, if we choose $k = 2, \alpha = 3$, then we have rates set like $\{\frac{3}{4}, \frac{3^2}{4^2}, \ldots, \frac{3^n}{4^n}\}$.

5.7.3 Analysis of Scalable Fuzzy Tracking

Security Analysis. For adversaries without holding secret keys, arguments for security are the same as standard encryption. We consider the unlinkability defined in Definition 5.3.12, then we argue it also satisfies unbiased fuzziness defined in Definition 5.3.13. Intuitively, unlinkability is to make true-positive and false-positive indistinguishable from the tracking server; And unbiased fuzziness is to make the hint' of each potential mpk' uniformly random for the sender.
Proof of Theorem 5.5.5 We restate the theorem for unlinkability formally here.

Theorem 5.7.8. *The fuzzy scalable stealth signature constructed in Figure 5.10 is unlinkable with key exposure and fuzzy tracking. Specifically, for any λ, N, ρ, if there is a PPT adversary $\mathcal{A}$ has non-negligible advantage in experiment defined in Figure 5.6, then there exist other adversaries $\mathcal{B}$ running in roughly same time such that:*

$$\mathsf{Adv}_{\lambda,N,\rho}^{UNLNK_{\mathsf{fs_{w-ke}}}}(\mathcal{A}) = \mathsf{Adv}_{\lambda}^{UNLNK_{\mathsf{w-ke}}}(\mathcal{B}) + \mathsf{Adv}_{\ell,q,\eta}^{\mathsf{MLWE}}(\mathcal{C}).$$

Proof. First, consider the two hybrids as follows:
Hybrid$_0$: This is the standard experiment.
Hybrid$_1$: This only changes ftki_b to ftki_{1-b} when $\mathsf{hint}_0 \in \mathsf{list} \wedge \mathsf{hint}_1 \in \mathsf{list}$ whereas $\mathsf{opk}_b, \mathsf{tki}_b$ remain unchanged.

Claim 5.7.2. Hybrid$_0$ *and* **Hybrid$_1$** *are computationally indistinguishable to the adversary if the decisional* MLWE *holds.*

Proof. Since we map each mpk to hint, we only need to consider the case where $\mathsf{hint}_0 \in \mathsf{list} \wedge \mathsf{hint}_1 \in \mathsf{list}$ as otherwise $b' \leftarrow\!\!\$ \; \{0,1\}$ and $\mathcal{A}_2$ will not be invoked. Without loss of generality, we assume $\mathsf{ftki}_b = \mathsf{ftki}_0$ and $\mathsf{hint}_0 = \mathsf{list}[i]$ which implies that, for list generated from ftki_0 and $\forall j \in [|\mathsf{list}|]$, there is

$$\begin{aligned}
w_0^j &= \lceil \mathbf{s}^T \mathbf{c}_1^j - c_2 \rfloor_2 \oplus y^j \\
&= \lceil \frac{q}{2}(s_1(x^i - x^j)) + e' - \frac{q}{2}(w_0 + y^i) \rfloor_2 \oplus y^j \\
&= \lceil \frac{q}{2}(s_1(x^i - x^j)) + e' - \frac{q}{2}(w_0) \rfloor_2 \oplus (y^i \oplus y^j) \\
&= \lceil \frac{q}{2}(w_0 + e' + s_1(x^i - x^j)) \rfloor_2 \oplus y^i \oplus y^j \\
&= w_0 \oplus (y^i \oplus y^j) \oplus \lceil \frac{q}{2}(s_1(x^i - x^j)) \rfloor_2,
\end{aligned}$$

where s_1 is the first ring element of $\mathbf{s}$ and $\mathsf{hint}_0 = \mathsf{decode}_{R_q}(w_0)[:\, n]$. On the other hand, if hint_1 (i.e., w_1) appears in the list with index k, i.e., $w_1 = \mathsf{list}[k] = w_0^k$, then the list can also be generated from ftki_1 because $\forall j \in [|\mathsf{list}|]$:

$$\begin{aligned}
w_1^j &= w_1 \oplus (y^k \oplus y^j) \oplus \lceil \frac{q}{2}(s_1(x^k - x^j)) \rfloor_2 \\
&= w_0^k \oplus (y^k \oplus y^j) \oplus \lceil \frac{q}{2}(s_1(x^k - x^j)) \rfloor_2 \\
&= w_0 \oplus (y^i \oplus y^k) \oplus \lceil \frac{q}{2}(s_1(x^i - x^k)) \rfloor_2 \\
&\quad \oplus (y^k \oplus y^j) \oplus \lceil \frac{q}{2}(s_1(x^k - x^j)) \rfloor_2 \\
&= w_0 \oplus (y^i \oplus y^j) \oplus \lceil \frac{q}{2}(s_1(x^i - x^j)) \rfloor_2 \\
&= w_0^j,
\end{aligned}$$

which means ftki_0 and ftki_1 will generate exactly the same list. Particularly, there is

$$\frac{q}{2}(w_0 + y^i + s_1 x^i) = \frac{q}{2}(w_1 + y^j + s_1 x^j).$$

Now consider the hybrids. We have $\mathsf{ftki}_0 := (\mathbf{c}_1, c_2)$ and $\mathsf{ftki}_1 := (\mathbf{c}_1', c_2')$, specifically,

$$\mathbf{c}_1 = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 + \frac{q}{2} \cdot \begin{bmatrix} x^i \\ 0 \end{bmatrix}, c_2 = \mathbf{b}^T\mathbf{r} + e_2 + \frac{q}{2} \cdot (w_0 + y^i)$$
$$\mathbf{c}_1' = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 + \frac{q}{2} \cdot \begin{bmatrix} x^j \\ 0 \end{bmatrix}, c_2' = \mathbf{b}^T\mathbf{r} + e_2 + \frac{q}{2} \cdot (w_1 + y^j).$$

Thus, there is

$$\begin{aligned}(\mathbf{c}_1, c_2) &\approx_c (\mathbf{u} + \frac{q}{2} \cdot \begin{bmatrix} x^i \\ 0 \end{bmatrix}, \mathbf{s}^T\mathbf{u} + \frac{q}{2} \cdot (w_0 + y^i) + e') \\ &\approx_s (\mathbf{u}', \mathbf{s}^T\mathbf{u}' + \frac{q}{2} \cdot (w_0 + y^i - \mathbf{s}^T \begin{bmatrix} x^i \\ 0 \end{bmatrix}) + e') \\ &= (\mathbf{u}', \mathbf{s}^T\mathbf{u}' + \frac{q}{2} \cdot (w_0 + y^i + s_1 x^i) + e') \\ &= (\mathbf{u}', \mathbf{s}^T\mathbf{u}' + \frac{q}{2} \cdot (w_1 + y^j + s_1 x^j) + e') \\ &\approx_c (\mathbf{c}_1', c_2'),\end{aligned}$$

where $e' = e_2 + \mathbf{e}^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1$ is the small noise term and $\mathbf{u}' = \mathbf{u} - \frac{q}{2} \cdot \begin{bmatrix} x^i \\ 0 \end{bmatrix}$. Note that $\frac{q}{2}(+s_1x^i) = \frac{q}{2}(-s_1x^i) \mod q$. Therefore, we have shown that ftki_0 and ftki_1 are indistinguishable for the adversary. □

Since ftki_b and ftki_{1-b} are indistinguishable and exchangeable for $\mathcal{A}$ in $\text{UNLNK}_{\mathsf{fs}_{\mathsf{w-ke}}}$. Now we show that $\mathcal{B}$ can fully simulate the $\text{UNLNK}_{\mathsf{fs}_{\mathsf{w-ke}}}$ experiment as follows. Upon receiving mpk_0, mpk_1, opk_b, tki_b, osk_b from $\text{UNLNK}_{\mathsf{w-ke}}$, $\mathcal{B}$ sample ftk, fpk then computes corresponding $\mathsf{ftki}_{b'}$ and list for $b' \leftarrow\$ \{0,1\}$, such that $w_0 \in \mathsf{list} \land w_1 \in \mathsf{list}$ where $\mathsf{H}(\mathsf{mpk}_0) = \mathsf{decode}_{R_q}(w_0)[: n]$ and $\mathsf{H}(\mathsf{mpk}_1) = \mathsf{decode}_{R_q}(w_1)[: n]$. Then $\mathcal{B}$ forwards all of them to $\mathcal{A}$ of $\text{UNLNK}_{\mathsf{fs}_{\mathsf{w-ke}}}$ and $\mathcal{A}$ cannot distinguish between ftki_0 or ftki_1 due to Claim 5.7.2. If $\mathcal{A}$ has non-negligible advantage $u(\lambda)$ in $\text{UNLNK}_{\mathsf{fs}_{\mathsf{w-ke}}}$, then $\mathcal{B}$ has the same non-negligible advantage $u(\lambda)$ in $\text{UNLNK}_{\mathsf{w-ke}}$. □

We restate the theorem for $\text{UNI-UBS}_{\mathsf{fs}}$ formally here.

Theorem 5.7.9. *If there is $n \leq \frac{m}{2}$ where m is a power of 2, and B_η is a centered binomial distribution, then the scalable fuzzy tracking constructed in Figure 5.10 is information theoretically* unbiased *and satisfies $\text{UNI-UBS}_{\mathsf{fs}}$ defined in Definition 5.3.13.*

Proof. If $\mathcal{A}$ is able to output valid mpk^i (i.e., valid hint^i and w^i), then for him, there is

$$w^j = w^i \oplus y^i \oplus y^j \oplus \lceil \frac{q}{2}(s_1(x^i - x^j)) \rfloor_2.$$

Note that the coefficients of $\lceil \frac{q}{2} s_1 \rfloor_2$ are uniformly random over $\{0,1\}^m$ because $s_1 \leftarrow\!\$\, B_\eta$ where B_η is a centred binomial distribution. Moreover, since polynomial multiplication can be written as circular convolution, $\lceil \frac{q}{2}(s_1(x^i - x^j)) \rfloor_2$ can be written as $\mathbf{Xs} \mod 2$ where $\mathbf{s} \leftarrow \mathsf{decode}_{R_q}(\lceil \frac{q}{2} s_1 \rfloor_2)$ and $\mathbf{X}$ is the circulant matrix represented by the polynomial $x \leftarrow \lceil \frac{q}{2}(x^i - x^j) \rfloor_2$. Specifically, the first column of $\mathbf{X}$ is $\mathsf{decode}_{R_q}(x)$ and other columns are rotational shifts of the previous column. Since m is a power of 2, it only has divisors from 2^0 to $2^{\log m}$. According to Lemma 1.0.2 and Definition 1.3.1, the biggest divisor of $X^m - 1$ is the polynomial $\Phi_m(X) = X^{\frac{m}{2}} + 1$ with degree $\frac{m}{2}$. Thus the rank of $\mathbf{X}$ is at least $m - \frac{m}{2}$ and at least a half of elements in $\mathbf{Xs} \mod 2$ are uniformly randomly distributed. This means $\mathsf{hint}^j \leftarrow \mathsf{decode}_{R_q}(w^j)[:n]$ is uniformly random as long as $n \leq \frac{m}{2}$. $\square$

6 Conclusion

This thesis is centred around enhancing efficiency and reducing the costs of communication and computation for commonly used privacy-preserving primitives, including private set intersection, oblivious transfer, and stealth signatures. Specifically, In Chapter 2, we present a protocol of multiparty threshold private set intersection, which improves communication bandwidth from $\tilde{O}(N^2t^2)$ to $\tilde{O}(Nt^2)$ where N is the number of parties and t the threshold while retaining the same computational overhead and security level. In Chapter 3, we introduce a new primitive, laconic private set intersection, which solves unbalanced PSI in a non-interactive way while making communication bandwidth as succinct as possible. Specifically, after the server publishes a short digest of constant size, any client can non-interactively send its message of size independent of the server's dataset. In Chapter 4, we present a two-message oblivious transfer protocol which has asymptotically minimum communicational bandwidth, namely, to transfer n bits information, it only requires $n(1+o(1))$ bits bandwidth for each user while retaining computational efficiency. We also show how to efficiently emulate $\mathbb{Z}_2$ inside a prime-order group $\mathbb{Z}_p$ in a function-private manner. In Chapter 5, we present a post-quantum privacy-preserving signature called stealth signature that saves 70% bandwidth compared to the state of the art while achieving the strongest security. Additionally, we present a fuzzy variant which protects users' metadata and improves the server's computational work from $O(N)$ to $O(\sqrt{N})$ where N is the number of users.

A
Additional Constructions

A.1 Threshold PSI: Oblivious Linear Algebra

A.1.1 Oblivious Matrix Multiplication

Protocol. The following Protocol 4 allows several parties to jointly compute the (encrypted) product of two encrypted matrices. Note that the protocol can also be used to compute the encryption of the product of two encrypted values in $\mathbb{F}$.

Analysis. We proceed to the analysis of the protocol described above.

Lemma A.1.1 (Correctness). *The protocol* $\mathsf{secMult}$ *is correct.*

Proof. The correctness is straightforward. $\square$

Lemma A.1.2 (Security). *The protocol* $\mathsf{secMult}$ *securely EUC-realizes* $\mathcal{F}_{\mathsf{OMM}}$ *with shared ideal functionality* $\mathcal{F}_{\mathsf{Gen}}$ *against semi-honest adversaries corrupting up to* $N-1$ *parties, given that* TPKE *is IND-CPA.*

Proof (Sketch). Assume that the adversary corrupts $N-k$ parties. The simulator takes the inputs from these parties and send them to the ideal functionality. Upon receiving the encrypted value $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l \cdot \mathbf{M}_r)$, it simulates the protocol as the honest parties would do.

We now prove that no set of at most $N-1$ colluding parties can extract information about $\mathbf{M}_l, \mathbf{M}_r$. First, observe that any set of $N-1$ parties cannot extract any information about encrypted values that are not decrypted during the protocol (because there is always a missing secret key share) given that TPKE is IND-CPA. Second, we analyze the matrix $\mathbf{M}'_l$ (which is decrypted during the protocol). We have that $\mathbf{M}'_l = \mathbf{M}_l + \sum_j \mathbf{R}_l^{(j)}$. Hence, there is always at least one matrix $\mathbf{R}_l^{(\ell)}$ which is unknown to the adversary and that perfectly hides the matrix $\mathbf{M}_l$ (the same happens $\mathbf{M}'_r$. $\square$

Algorithm 4 Secure Multiplication secMult

```
Require: Each party $\mathsf{P}_i$ has a secret share $\mathsf{sk}_i$ of a secret key for a public key $\mathsf{pk}$ of a TPKE
    scheme $\mathsf{TPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
Ensure: Party $\mathsf{P}_1$ inputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l)$ and $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}_r)$, where $\mathbf{M}_l, \mathbf{M}_r \in \mathbb{F}^{t\times t}$.
    Goal: Every one knows the product $\mathsf{Enc}(\mathbf{M}_l \cdot \mathbf{M}_r)$.
1:  for all party $\mathsf{P}_i$ do
2:      It samples two random matrices $\mathbf{R}_l^{(i)}, \mathbf{R}_r^{(i)} \leftarrow\$ \mathbb{F}^{t\times t}$.
3:      It computes $c_l^{(i)} = \mathsf{Enc}(\mathsf{pk}, \mathbf{R}_l^{(i)})$, $c_l^{(i)} = \mathsf{Enc}(\mathsf{pk}, \mathbf{R}_r^{(i)})$, $d_r^{(i)} = \mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l \cdot \mathbf{R}_r^{(i)})$,
        $d_l^{(i)} = \mathsf{Enc}(\mathsf{pk}, \mathbf{R}_l^{(i)} \cdot \mathbf{M}_r)$.
4:      It broadcasts $\{c_l^{(i)}, c_r^{(i)}, d_l^{(i)}, d_r^{(i)}\}$.
5:  end for
6:  Each party $\mathsf{P}_i$ computes $\tilde{c}^{(i)} = \mathsf{Enc}(\mathsf{pk}, \sum_{j\neq i} \mathbf{R}_l^{(i)} \cdot \mathbf{R}_r^{(j)})$ (using $c_r^{(j)}$ and $\mathbf{R}_l^{(i)}$) and broad-
    casts $\tilde{c}^{(i)}$.
7:  All parties mutually decrypt i) $\mathsf{Enc}(\mathbf{M}'_l) := \mathsf{Enc}(\mathsf{pk}, \mathbf{M}_l) + \sum_j c_l^{(j)}$ (to obtain $\mathbf{M}'_l \in$
    $\mathbb{F}^{t\times t}$), ii) $\mathsf{Enc}(\mathbf{M}'_r) := \mathsf{Enc}(\mathsf{pk}, \mathbf{M}_r) + \sum_j c_r^{(j)}$ (to obtain $\mathbf{M}'_r \in \mathbb{F}^{t\times t}$)
8:  for all party $\mathsf{P}_i$ do
9:      It computes $\tilde{d} = \mathsf{Enc}(\mathsf{pk}, \mathbf{M}'_l \cdot \mathbf{M}'_r)$.
10:     It outputs $e = \tilde{d} - \sum_j d_l^{(j)} - \sum_j d_r^{(j)} - \sum_j \tilde{c}^{(j)}$
11: end for
```

Complexity. The communication complexity of the protocol is dominated by the messages carrying the (encrypted) matrix. Hence, assuming a broadcast channel between the parties, the protocol has communication complexity of $\mathcal{O}(Nt^2)$ where t is the size of the input matrices and N the number of parties involved in the protocol.

A.1.2 Compute the Rank of a Matrix

Protocol. We now present the Protocol 5 to compute the rank of an encrypted matrix.

```
Algorithm 5 Secure Rank secRank
Require: Each party has a secret key share $\mathsf{sk}_i$ for a public key $\mathsf{pk}$ of a TPKE $\mathsf{TPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The parties have access to the oblivious matrix multiplication ideal functionality $\mathcal{F}_{\mathsf{OMM}}$.
Ensure: Party $\mathsf{P}_1$ inputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{M})$ where $\mathbf{M} \in \mathbb{F}^{t \times t}$.
1: Each party $\mathsf{P}_i$ broadcasts an encrypted uniformly chosen at random unit upper and lower triangular Toeplitz matrices $\mathsf{Enc}(\mathsf{pk}, \mathbf{U}_i)$ and $\mathsf{Enc}(\mathsf{pk}, \mathbf{Z}_i)$ and a uniformly chosen at random diagonal matrix $\mathsf{Enc}(\mathsf{pk}, \mathbf{X}_i)$, where $\mathbf{U}_i, \mathbf{Z}_i \in \mathbb{F}^{t \times t}$ and $\mathbf{X}_i \in \mathbb{F}^{t \times t}$.
2: Each party $\mathsf{P}_i$ computes: i) $\mathsf{Enc}(\mathsf{pk}, \mathbf{X}) = \sum_i \mathsf{Enc}(\mathsf{pk}, \mathbf{X}_i)$, ii) $\mathsf{Enc}(\mathsf{pk}, \mathbf{U}) = \mathsf{Enc}(\mathsf{pk}, (\sum_i \mathbf{U}_i) - (N-1)\mathbf{I})$, and iii) $\mathsf{Enc}(\mathsf{pk}, \mathbf{Z}) = \mathsf{Enc}(\mathsf{pk}, (\sum_i \mathbf{Z}_i) - (N-1)\mathbf{I})$, where $\mathbf{I}$ is the identity matrix.
3: All parties mutually compute $\mathsf{Enc}(\mathsf{pk}, \mathbf{N}) = \mathsf{Enc}(\mathsf{pk}, \mathbf{XUMZ})$ via three invocations of $\mathcal{F}_{\mathsf{OMM}}$.
4: Each party $\mathsf{P}_i$ samples $\mathbf{u}_i, \mathbf{v}_i \leftarrow\$ \mathbb{F}^t$ and broadcasts $\mathsf{Enc}(\mathsf{pk}, \mathbf{u}_i)$, $\mathsf{Enc}(\mathsf{pk}, \mathbf{v}_i)$.
5: Each party $\mathsf{P}_i$ computes $\mathsf{Enc}(\mathsf{pk}, \mathbf{u}) = \sum_j \mathsf{Enc}(\mathsf{pk}, \mathbf{u}_j)$ and $\mathsf{Enc}(\mathsf{pk}, \mathbf{v}) = \sum_j \mathsf{Enc}(\mathsf{pk}, \mathbf{v}_j)$. Then, it computes the sequence $\mathsf{Enc}(\mathfrak{a})$ with $2\log t$ invocations of $\mathcal{F}_{\mathsf{OMM}}$,[1] where $\mathfrak{a} = \{\mathbf{a}_0, \ldots, \mathbf{a}_{2t-1}\}$ and $\mathsf{Enc}(\mathsf{pk}, \mathbf{a}_j) = \mathsf{Enc}(\mathsf{pk}, \mathbf{u}\mathbf{N}^j\mathbf{v})$ for $0 \leq j \leq 2t-1$.
6: All parties mutually compute $\mathsf{Enc}(\mathsf{pk}, r-1)$ where $r$ is the degree of $m_{\mathfrak{a}}$, the minimal polynomial of the (encrypted) sequence $\mathsf{Enc}(\mathfrak{a})$. This can be calculated using a Boolean circuit with size $O(t^2 k \log t)$ (which can be securely constructed from TPKE [ST06]).
```

Analysis. We analyze the correctness and security of the protocol.

Lemma A.1.3 (Correctness). *The protocol* secRank *is correct.*

Proof. The correctness of the protocol is guaranteed by Lemma 1.2.2 and Lemma 1.2.3. □

[1] We can perform t multiplications in $\mathcal{O}(\log t)$ calls to $\mathcal{F}_{\mathsf{OMM}}$ by performing multiplications in a batched fashion [KMWF07].

Lemma A.1.4 (Security). *The protocol* $\mathsf{secRank}$ *securely EUC-realizes* $\mathcal{F}_{\mathsf{ORank}}$ *with shared ideal functionality* $\mathcal{F}_{\mathsf{Gen}}$ *in the* $\mathcal{F}_{\mathsf{OMM}}$*-hybrid model against semi-honest adversaries corrupting up to* $N-1$ *parties, given that* TPKE *is IND-CPA.*

Proof (Sketch). The simulator takes the corrupted parties input, sends them to the ideal functionality and simulates the protocol as the honest parties would do. It is easy to see that, even when the adversary corrupts $N-1$ parties, the information is hidden by the TPKE and thus no information on $\mathbf{M}$ is leaked to the adversary by the IND-CPA of the underlying TPKE. □

Complexity. Each party broadcasts $\mathcal{O}(t^2 k \log t)$ bits of information, where $k = \log |\mathbb{F}|$. To see this, note that the communication of the protocol is dominated by the computation of the circuit that computes the degree of $\mathfrak{a}$ and this can be implemented with communication cost of $\mathcal{O}(t^2 k \log t)$ [KMWF07]. Assuming a broadcast channel, the communication complexity is $\tilde{\mathcal{O}}(Nt^2)$

A.1.3 Invert a Matrix

In this section, we present and analyze a protocol that allows N parties to invert an encrypted matrix. In this setting, each of the N parties holds a secret share of a public key pk of a TPKE. Given an encrypted matrix, they want to compute an encryption of the inverse of this matrix.

Ideal Functionality. The ideal functionality of oblivious rank computation is defined below.

$\mathcal{F}_{\mathsf{OInv}}$ **functionality**

Parameters: $\mathsf{sid}, N, q, t \in \mathbb{N}$ and $\mathbb{F}$, where $\mathbb{F}$ is a field of order q, known to the N parties involved in the protocol. pk public-key of a threshold PKE scheme.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_1, \mathsf{Enc}(\mathsf{pk}, \mathbf{M}))$ from party P_1 (where $\mathbf{M} \in \mathbb{F}^{t\times t}$ is a non-singular matrix), $\mathcal{F}_{\mathsf{ORank}}$ outputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}^{-1})$ to P_1 and $(\mathsf{Enc}(\mathsf{pk}, \mathbf{M}), \mathsf{Enc}(\mathsf{pk}, \mathbf{M}^{-1}))$ to all other parties P_i, for $i = 2, \ldots, N$.

Protocol. We now describe the Protocol 6 that allows N parties to jointly compute the encryption of the inverse of a matrix, given that this matrix is non-singular.

Analysis. The proofs of the following lemmas follow the same lines as the proofs in the analysis of $\mathsf{secMult}$ protocol. We state the lemmas but omit the proofs for briefness.

Lemma A.1.5. *The protocol* secInv *is correct.*

Lemma A.1.6. *The protocol* secInv *securely EUC-realizes* $\mathcal{F}_{\mathsf{OInv}}$ *with shared ideal functionality* $\mathcal{F}_{\mathsf{Gen}}$ *against semi-honest adversaries corrupting up to* $N-1$ *parties, given that* TPKE *is IND-CPA.*

Algorithm 6 Secure Matrix Invert secInv

```
Require: Each party has a secret key share $\mathsf{sk}_i$ for a public key $\mathsf{pk}$ of a TPKE $\mathsf{TPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
Ensure: Party $\mathsf{P}_1$ inputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{M})$ where $\mathbf{M} \in \mathbb{F}^{t \times t}$ is a non-singular matrix.
 1: Each party $\mathsf{P}_i$ samples a non-singular matrix $\mathbf{R}_i \leftarrow\$ \mathbb{F}^{t \times t}$.
 2: Set $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}') := \mathsf{Enc}(\mathsf{pk}, \mathbf{M})$.
 3: for $i$ from 1 to $N$ do
 4:     $\mathsf{P}_i$ calculates $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}') = \mathsf{Enc}(\mathsf{pk}, \mathbf{R}_i\mathbf{M}')$
 5:     $\mathsf{P}_i$ broadcasts $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}')$.
 6: end for
 7: All parties mutually decrypt the final $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}')$. Then they compute its inverse to
    obtain $\mathsf{Enc}(\mathsf{pk}, \mathbf{N}') = \mathsf{Enc}(\mathsf{pk}, \mathbf{M}'^{-1} \prod_i \mathbf{R}_i^{-1})$.
 8: for $i$ from $N$ to 1 do
 9:     $\mathsf{P}_i$ computes $\mathsf{Enc}(\mathsf{pk}, \mathbf{N}') = \mathsf{Enc}(\mathsf{pk}, \mathbf{N}'\mathbf{R}_i^{-1})$.
10:     $\mathsf{P}_i$ broadcasts $\mathsf{Enc}(\mathsf{pk}, \mathbf{N}')$
11: end for
12: Finally, $\mathsf{P}_1$ outputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}^{-1}) = \mathsf{Enc}(\mathsf{pk}, \mathbf{N}')$.
```

Complexity. Each party broadcasts $\mathcal{O}(t^2)$ bits of information. The communication complexity of the protocol is $\mathcal{O}(Nt^2)$, assuming a broadcast channel.

A.1.4 Secure Unary Representation

Following [KMWF07], we present a protocol that allows to securely compute the unary representation of a matrix.

Ideal Functionality. The ideal functionality for Secure Unary Representation is given below.

$\mathcal{F}_{\mathsf{SUR}}$ functionality

Parameters: $\mathsf{sid}, N, q, t \in \mathbb{N}$ and $\mathbb{F}$, where $\mathbb{F}$ is a field of order q, known to the N parties involved in the protocol. pk public-key of a threshold PKE scheme.

- Upon receiving $(\mathsf{sid}, \mathsf{P}_1, \mathsf{Enc}(\mathsf{pk}, r))$ from party P_1 (where $r \in \mathbb{F}$ and $r \leq t$), $\mathcal{F}_{\mathsf{SUR}}$ computes $(\mathsf{Enc}(\mathsf{pk}, \delta_1), \ldots, \mathsf{Enc}(\mathsf{pk}, \delta_t))$ such that $\delta_i = 1$ if $i \leq r$, and $\delta_i = 0$ otherwise. The functionality outputs $(\mathsf{Enc}(\mathsf{pk}, \delta_1), \ldots, \mathsf{Enc}(\mathsf{pk}, \delta_t))$ to P_1 and $(\mathsf{Enc}(\mathsf{pk}, r), (\mathsf{Enc}(\mathsf{pk}, \delta_1), \ldots, \mathsf{Enc}(\mathsf{pk}, \delta_t)))$ to all other parties P_i, for $i = 2, \ldots, N$.

Protocol. A protocol for secure unary representation can be implemented with the help of a binary-conversion protocol [ST06]. That is, given $\mathsf{Enc}(\mathsf{pk}, r)$, all parties jointly compute $\mathsf{Enc}(\mathsf{pk}, \delta_i)$, where $\delta_i = 1$, if $i \leq r$, and $\delta_i = 0$ otherwise, via a Boolean circuit (which can be securely implemented based on Paillier cryptosystem).

Communication complexity. We can calculate the result using a Boolean circuit of size $O(r \log t)$, thus the communication complexity is $O(Nr \log t)$.

A.1.5 Solve a Linear System

Protocol. We now present the Protocol 7 that allows multiple parties to solve an encrypted linear system. In the following, we assume that the system has at least one solution (note that this can be guaranteed using the $\mathsf{secRank}$ protocol).

Lemma A.1.7 (Correctness). *The protocol* secLS *is correct.*

Proof. The proof follows directly from [KDS91, KMWF07]. $\square$

Lemma A.1.8. *The protocol* secLS *securely EUC-realizes* $\mathcal{F}_{\mathsf{OLS}}$ *with shared ideal functionality* $\mathcal{F}_{\mathsf{Gen}}$ *in the* $(\mathcal{F}_{\mathsf{ORank}}, \mathcal{F}_{\mathsf{OInv}}, \mathcal{F}_{\mathsf{SUR}})$*-hybrid model against semi-honest adversaries corrupting up to* $N-1$ *parties, given that* TPKE *is IND-CPA.*

Communication complexity. Each party broadcasts $\mathcal{O}(t^2 k \log t)$ bits of information where $k = |\mathbb{F}|$. The total communication complexity is $\tilde{\mathcal{O}}(t^2)$.

A.2 Stealth Signature: Group-based Construction against Bounded Leakage

We provide an SS which is unforgeable and unlinkable with *bounded* key-exposure. The construction is shown in Figure A.1, where $\mathbb{G}$ is a group of primer order p, g is a generator, and H is a random oracle mapping from $\mathbb{G}$ to $\mathbb{Z}_p$. Additionally, DS is an efficient group-based signature scheme such as ECDSA, Schnorr and others whose verification key and signing key has discrete logarithm relation, i.e., $\mathsf{vk} = g^{\mathsf{sk}}$.

Correctness. It is clear that $\mathsf{opk} = g^{\mathsf{osk}}$ as

$$\mathsf{opk} = \prod_{i=1}^{n} h_i^{\mathsf{H}(h_i^r)} = g^{\sum_{i=1}^{n} x_i \cdot \mathsf{H}(g^{r \cdot x_i})} = g^{\mathsf{osk}}.$$

The tracking mechanism also works since

$$R_0 = \mathsf{opk}^{\mathsf{H}(h_0^r)} = \mathsf{opk}^{\mathsf{H}(g^{r \cdot x_0})}.$$

Algorithm 7 Secure Linear Solve secLS

```
Require: Each party has a secret key share $\mathsf{sk}_i$ for a public key $\mathsf{pk}$ of a TPKE $\mathsf{TPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The parties have access to the ideal functionalities $\mathcal{F}_{\mathsf{ORank}}$, $\mathcal{F}_{\mathsf{OInv}}$ and $\mathcal{F}_{\mathsf{SUR}}$.
Ensure: Party $\mathsf{P}_1$ inputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{M})$ where $\mathbf{M} \in \mathbb{F}^{t \times t}$ is a non-singular matrix.
1:  All parties jointly compute an encryption of the rank $\mathsf{Enc}(\mathsf{pk}, r)$ of $\mathbf{M}$ via the ideal functionality $\mathcal{F}_{\mathsf{ORank}}$.
2:  Set $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}') := \mathsf{Enc}(\mathsf{pk}, \mathbf{M})$ and $\mathsf{Enc}(\mathsf{pk}, \mathbf{y}') := \mathsf{Enc}(\mathsf{pk}, \mathbf{y})$.
3:  for $i$ from $1$ to $N$ do
4:      $\mathsf{P}_i$ samples two non-singular matrices $\mathbf{R}_i, \mathbf{Q}_i$ from $\mathbb{F}^{t \times t}$. It calculates $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}') = \mathsf{Enc}(\mathsf{pk}, \mathbf{R}_i \mathbf{M}' \mathbf{Q}_i)$ and $\mathsf{Enc}(\mathsf{pk}, \mathbf{y}') = \mathsf{Enc}(\mathsf{pk}, \mathbf{R}_i \mathbf{y}')$. $\mathsf{P}_i$ broadcasts $\mathsf{Enc}(\mathsf{pk}, \mathbf{M}')$, $\mathsf{Enc}(\mathsf{pk}, \mathbf{y}')$.
5:  end for
6:  All the parties jointly compute $\mathsf{Enc}(\delta_1), \ldots, \mathsf{Enc}(\delta_t)$ by invoking $\mathcal{F}_{\mathsf{SUR}}$ on input $\mathsf{Enc}(\mathsf{pk}, r)$. They set $\mathsf{Enc}(\mathsf{pk}, \Delta) := \mathsf{Enc}\left(\mathsf{pk}, \begin{bmatrix} \delta_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \delta_t \end{bmatrix}\right)$. Finally, they compute $\mathsf{Enc}(\mathsf{pk}, \mathbf{N}) := \mathsf{Enc}(\mathsf{pk}, \mathbf{M}' \cdot \Delta + \mathbf{I}_t - \Delta)$, where $\mathbf{I}_t \in \mathbb{F}^{t \times t}$ is the identity matrix.
7:  All the parties jointly compute $\mathsf{Enc}(\mathbf{N}^{-1})$ by invoking $\mathcal{F}_{\mathsf{OInv}}$ on input $\mathsf{Enc}(\mathsf{pk}, \mathbf{N})$.
8:  Each party $\mathsf{P}^i$ samples $\mathbf{u}_i \leftarrow\$ \mathbb{F}^t$ and broadcasts $(\mathsf{Enc}(\mathsf{pk}, \mathbf{M}'\mathbf{u}_i), \mathsf{Enc}(\mathsf{pk}, \mathbf{u}_i))$.
9:  All parties jointly compute $\mathsf{Enc}(\mathsf{pk}, \mathbf{u}') = \mathsf{Enc}(\mathsf{pk}, \mathbf{N}^{-1}\mathbf{y}'_r)$ by invoking $\mathcal{F}_{\mathsf{OMM}}$, where $\mathsf{Enc}(\mathsf{pk}, \mathbf{y}'_r) = \mathsf{Enc}(\mathsf{pk}, (\mathbf{y}' + \sum_j \mathbf{M}'\mathbf{u}_j)\Delta)$. Then they set $\mathsf{Enc}(\mathsf{pk}, \mathbf{x}) = \mathsf{Enc}(\mathsf{pk}, (\sum_j \mathbf{u}_j) - \mathbf{u}')$.
10: for $i$ from $N$ to $1$ do
11:     $\mathsf{P}_i$ calculates $\mathsf{Enc}(\mathsf{pk}, \mathbf{x}) = \mathsf{Enc}(\mathsf{pk}, \mathbf{Q}_i^{-1}\mathbf{x})$. $\mathsf{P}_i$ broadcasts $\mathsf{Enc}(\mathsf{pk}, \mathbf{x})$.
12: end for
13: $\mathsf{P}_1$ outputs $\mathsf{Enc}(\mathsf{pk}, \mathbf{x})$.
```

```
$\mathsf{MKGen}(\lambda)$
Sample $\mathbb{G} = \langle g \rangle$
$(x_0, x_1, \dots, x_n) \leftarrow\!\$\ \mathbb{Z}_p$
$\mathsf{mpk} := (g, h_0 := g^{x_0}, \dots, h_n := g^{x_n})$
$\mathsf{msk} := (x_0, \dots, x_n)$
$\mathsf{mtk} := x_0$
return mpk, msk, mtk
```

```
$\mathsf{OPKGen}(\mathsf{mpk})$
parse $(g, h_0, \dots, h_n) := \mathsf{mpk}$
$r \leftarrow\!\$\ \mathbb{Z}_p$
$\mathsf{opk} := \prod_{i=1}^{n} h_i^{\mathsf{H}(h_i^r)}$
$\mathsf{tki} := (R := g^r, R_0 := \mathsf{opk}^{\mathsf{H}(h_0^r)})$
return opk, tki
```

```
$\mathsf{OSKGen}(\mathsf{msk}, \mathsf{opk}, \mathsf{tki})$
parse $(x_0, \dots, x_n) := \mathsf{msk}$
parse $(R, R_0) := \mathsf{tki}$
return $\perp$ if
false $\leftarrow \mathsf{Track}(x_0, , \mathsf{opk}, \mathsf{tki})$
$\mathsf{osk} := \sum_{i=1}^{n} x_i \cdot \mathsf{H}(R^{x_i})$
return osk
```

```
$\mathsf{Sign}(\mathsf{osk}, m)$
return $\sigma := \mathsf{DS.Sign}(\mathsf{osk}, m)$
```

```
$\mathsf{Track}(\mathsf{mtk}, \mathsf{opk}, \mathsf{tki})$
parse $(R, R_0) := \mathsf{tki}$
parse $x_0 := \mathsf{mtk}$
return $\mathsf{opk}^{\mathsf{H}(R^{x_0})} \stackrel{?}{=} R_0$
```

```
$\mathsf{Vf}(\mathsf{opk}, \sigma, m)$
return $\mathsf{DS.Vf}(\mathsf{opk}, \sigma, m)$
```

Figure A.1: Construction of group-based SS secure with $(n-1)$-bounded key-exposure

Security Analysis. Now we analyze the security of above construction.

Theorem A.2.1. *The construction in Figure A.1 is (strongly) unforgeable and unlinkable with* $(n-1)$-bounded *key exposures.*

Proof. (sketch) For unlinkability, without knowing x_i or r, by DDH assumption, the triple $g^r, g^{x_i}, g^{r \cdot x_i}$ remains uniformly random over $\mathbb{G}$. With random oracle H, $\mathsf{H}(g^{r \cdot x_i})$ is also uniformly random over $\mathbb{Z}_p$. Therefore, it is clear that opk, R, R_0 are uniformly random.

For unforgeability, as long as DS is (strongly) unforgeable, then SS is also (strongly) unforgeable.

Now we consider key-exposures. Since

$$\mathsf{osk} = \sum_{i=1}^{n} x_i \cdot \mathsf{H}(h_i^r),$$

this is an equation with n variables (x_i) for adversaries. If the adversary learns at most $n-1$ equations, then this linear system is undetermined and has at least p solutions which is exponentially large. Thus msk is hiding when there are at most $(n-1)$ key-exposures. □

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[ABD$^+$21a] Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 94–125, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany.

[ABD$^+$21b] Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. Cryptology ePrint Archive, Report 2021/728, 2021. https://ia.cr/2021/728.

[ADT11] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (If) size matters: Size-hiding private set intersection. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 156–173, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.

[AFLT12] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 572–590, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 120–129, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[Ajt98] Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *30th Annual ACM Symposium on Theory of Computing*, pages 10–19, Dallas, TX, USA, May 23–26, 1998. ACM Press.

[AMKM21] Lukas Aumayr, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Blitz: Secure multi-hop payments without two-phase commits. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 4043–4060. USENIX Association, August 11–13, 2021.

[AR16] Divesh Aggarwal and Oded Regev. A note on discrete gaussian combinations of lattice vectors. *Chicago Journal of Theoretical Computer Science*, 2016(7), June 2016.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–636, 1993.

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[BBC$^+$18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 669–699, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[BBD$^+$20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 58–87, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[BBV$^+$20] Alex Berke, Michiel Bakker, Praneeth Vepakomma, Kent Larson, and Alex 'Sandy' Pentland. Assessing disease exposure risk with location data: A proposal for cryptographic preservation of privacy, 2020.

[BCG$^+$19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 291–308, London, UK, November 11–15, 2019. ACM Press.

[BCG$^+$19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 407–437, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[BDM22] Pedro Branco, Nico Döttling, and Paulo Mateus. Two-round oblivious linear evaluation from learning with errors. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 379–408, Cham, 2022. Springer International Publishing.

[BdMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 62–89, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BdV18] Niek J. Bouman and Niels de Vreede. New protocols for secure linear algebra: Pivoting-free elimination and fast block-recursive matrix decomposition. *IACR Cryptology ePrint Archive*, 2018:703, 2018.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology*

– *CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BLMG21] Gabrielle Beck, Julia Len, Ian Miers, and Matthew Green. Fuzzy message detection. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 1507–1528, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BMRR21] Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. Multi-party threshold private set intersection with sublinear communication. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 349–379, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Com-*

puter Science, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CD01] Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 119–136, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[CDI$^+$19] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 462–488, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[CGH$^+$21] Melissa Chase, Sanjam Garg, Mohammad Hajiabadi, Jialin Li, and Peihan Miao. Amortizing rate-1 OT and applications to PIR and PSI. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 126–156, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press.

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1223–1237, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1243–1255, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[CM17] Nicolas T Courtois and Rebekah Mercer. Stealth address and key management techniques in blockchain systems. *ICISSP*, 2017:559–566, 2017.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 789–800, Berlin, Germany, November 4–8, 2013. ACM Press.

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 372–408, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[DGGM19] Nico Döttling, Sanjam Garg, Vipul Goyal, and Giulio Malavolta. Laconic conditional disclosure of secrets and applications. In David Zuckerman, editor, *60th Annual Symposium on Foundations of Computer Science*, pages 661–685, Baltimore, MD, USA, November 9–12, 2019. IEEE Computer Society Press.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 3–31, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

[DGI$^+$19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany.

[DKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 213–231, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

[DMRY09] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09: 7th International Conference on Applied Cryptography and Network Security*, volume 5536 of *Lecture Notes in Computer Science*, pages 125–142, Paris-Rocquencourt, France, June 2–5, 2009. Springer, Heidelberg, Germany.

[Döt15] Nico Döttling. Low noise LPN: KDM secure public key encryption and sample amplification. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 604–626, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[FT14] Pierre-Alain Fouque and Mehdi Tibouchi. Close to uniform prime number generation with fewer random bits. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014: 41st International Colloquium on Automata, Languages and Programming, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 991–1002, Copenhagen, Denmark, July 8–11, 2014. Springer, Heidelberg, Germany.

[GGH19] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 33–63, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[GHM$^+$19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 63–93, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany.

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 689–718, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[GHO20] Sanjam Garg, Mohammad Hajiabadi, and Rafail Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 OT and more. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 88–116, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

[GJR10] Elena Grigorescu, Kyomin Jung, and Ronitt Rubinfeld. A local decision test for sparse polynomials. *Inf. Process. Lett.*, 110(20):898–901, September 2010.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press.

[GMP22] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 402–432, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

[GMPW20] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. Improved discrete gaussian and subgaussian analysis for lattice cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 623–651, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

[GMW19] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. 2019.

[GN19] Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 154–185, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[GNN17] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 629–659, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[Gre19] Matthew Green, 2019. https://blog.cryptographyengineering.com/2019/12/08/on-client-side-media-scanning/.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 588–599, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[GS19a] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 3–29, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[GS19b] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. Cryptology ePrint Archive, Report 2019/175, 2019. `https://eprint.iacr.org/2019/175`.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 621–651, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

[GVW20] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. New constructions of hinting PRGs, OWFs with encryption, and more. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 527–558, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

[HOS17] Per A. Hallgren, Claudio Orlandi, and Andrei Sabelfeld. PrivatePool: Privacy-preserving ridesharing. In Boris Köpf and Steve Chong, editors, *CSF 2017: IEEE 30st Computer Security Foundations Symposium*, pages 276–291, Santa Barbara, CA, USA, August 21–25, 2017. IEEE Computer Society Press.

[HV17] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 175–203, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172, Rehovot, Israel, January 11–13, 2015. Association for Computing Machinery.

[IKN$^+$17] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive, Report 2017/738, 2017. https://eprint.iacr.org/2017/738.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[Ing56] A. W. Ingleton. The rank of circulant matrices. *Journal of the London Mathematical Society*, s1-31(4):445–460, 1956.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 294–314. Springer, Heidelberg, Germany, March 15–17, 2009.

[JL10] Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 418–435, Amalfi, Italy, September 13–15, 2010. Springer, Heidelberg, Germany.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 60–73, New York, NY, USA, 2021. Association for Computing Machinery.

[KDS91] Erich Kaltofen and B. David Saunders. On wiedemann's method of solving sparse linear systems. In Harold F. Mattson, Teo Mora, and T. R. N. Rao, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 29–38, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 818–829, Vienna, Austria, October 24–28, 2016. ACM Press.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 552–586, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[KMP$^+$17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1257–1272, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[KMWF07] Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew K. Franklin. Secure linear algebra using linearly recurrent sequences. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 291–310, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 669–684, Berlin, Germany, November 4–8, 2013. ACM Press.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[LDK+20] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.

[LLN+20] Wenling Liu, Zhen Liu, Khoa Nguyen, Guomin Yang, and Yu Yu. A lattice-based key-insulated and privacy-preserving signature scheme with publicly derived public key. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020: 25th European Symposium on Research in Computer Security, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 357–377, Guildford, UK, September 14–18, 2020. Springer, Heidelberg, Germany.

[LNO13] Yehuda Lindell, Kobbi Nissim, and Claudio Orlandi. Hiding the input-size in secure two-party computation. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 421–440, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[LRR+19] Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. Omniring: Scaling private payments without trusted setup. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 31–48, London, UK, November 11–15, 2019. ACM Press.

[LT22] Zeyu Liu and Eran Tromer. Oblivious message retrieval. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 753–783, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[LYW$^+$19] Zhen Liu, Guomin Yang, Duncan S. Wong, Khoa Nguyen, and Huaxiong Wang. Key-Insulated and Privacy-Preserving signature scheme with publicly derived public key. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 215–230, 2019.

[Mea86] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134, 1986.

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[MSH$^+$17] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *arXiv preprint arXiv:1704.04299*, 2017.

[MSS$^+$22] Varun Madathil, Alessandra Scafuro, István András Seres, Omer Shlomovits, and Denis Varlakov. Private signaling. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3309–3326, Boston, MA, August 2022. USENIX Association.

[MTZ03] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Trans. Information Theory*, 49(9):2213–2218, 2003.

[NMRL16] Shen Noether, Adam Mackenzie, and the Monero Research Lab. Ring confidential transactions. *Ledger*, 1:1–18, Dec. 2016.

[NW06] Kobbi Nissim and Enav Weinreb. Communication efficient secure linear algebra. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 522–541, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany.

[OKH13] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 401–431, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[PRV12] Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012. https://eprint.iacr.org/2012/653.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015: 24th USENIX Security Symposium*, pages 515–530, Washington, DC, USA, August 12–14, 2015. USENIX Association.

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 125–157, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014: 23rd USENIX Security Symposium*, pages 797–812, San Diego, CA, USA, August 20–22, 2014. USENIX Association.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 859–870, Paris, France, October 7–9, 2018. IEEE Computer Society Press.

[RA18] Amanda C. Davi Resende and Diego F. Aranha. Faster unbalanced private set intersection. In Sarah Meiklejohn and Kazue Sako, editors, *FC 2018: 22nd International Conference on Financial Cryptography and Data Security*, volume 10957 of *Lecture Notes in Computer Science*, pages 203–221, Nieuwpoort, Curaçao, February 26 – March 2, 2018. Springer, Heidelberg, Germany.

[Rab05] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptol. ePrint Arch.*, 2005(187), 2005.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[RH13] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[RR17a] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 235–259, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[RR17b] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1229–1242, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[RS13] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[SAB$^+$20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[SGS21] Gili Schul-Ganz and Gil Segev. Generic-group identity-based encryption: A tight impossibility result. Cryptology ePrint Archive, Report 2021/745, 2021. https://eprint.iacr.org/2021/745.

[SO13] Marc Santamaria Ortega. The bitcoin transaction graph anonymity. 2013.

[SPB21] István András Seres, Balázs Pejó, and Péter Burcsi. The effect of false positives: Why fuzzy message detection leads to fuzzy privacy guarantees? Cryptology ePrint Archive, Report 2021/1180, 2021. https://eprint.iacr.org/2021/1180.

[ST06] Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for Paillier encrypted values. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 522–537, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[TBM+20] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. Verifiable timed signatures made practical. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1733–1750, Virtual Event, USA, November 9–13, 2020. ACM Press.

[Tod] Peter Todd. Stealth addresses, 2014. http://www.mailarchive.com/bitcoin-development@lists.sourceforge.net/msg03613.html.

[vS] Nicolas van Saberhagen. Cryponote v 2.0. 2013. https://cryptonote.org/whitepaper.pdf.

[Weba] Website. https://anonymous.4open.science/r/SPIRIT-2CE8.

[Webb] Website. Cryptocurrency and e-commerce. https://shiphero.com/blog/cryptocurrency-e-commerce/.

[Webc] Website. Cryptocurrency and online gaming. https://forumpay.com/gaming.

[Webd] Website. Cryptocurrency solutions for institutional philanthropy. https://thegivingblock.com.

[Webe] Website. Digital currency donations for freedom convoy evading seizure by authorities. https://www.cbc.ca/news/canada/ottawa/freedom-convoy-cryptocurrency-asset-seizure-1.6389601.

[Webf] Website. How many people own and use bitcoin? https://www.buybitcoinworldwide.com/how-many-bitcoin-users/.

[Webg] Website. How to donate crypto. https://www.coinbase.com/learn/crypto-basics/how-to-donate-crypto.

[Webh] Website. Umbra: Privacy preserving stealth payments. https://gitcoin.co/grants/821/umbra-privacy-preserving-stealth-payments.

[Webi] Website. Untraceable transactions which can contain a secure message are inevitable. 2011. https://bitcointalk.org/index.php?topic=5965.0.

[Webj] Website. Why donate bitcoin, ethereum, nfts and other cryptocurrencies to charity.

[ZMS+21] Raymond K. Zhao, Sarah McCarthy, Ron Steinfeld, Amin Sakzad, and Máire O'Neill. Quantum-safe HIBE: does it cost a latte? Cryptology ePrint Archive, Report 2021/222, 2021. https://eprint.iacr.org/2021/222.