Lattice-based Accumulators and Applications

Chuanwei Lin, 202003408

Master's Thesis, Computer Science

June 2022

Advisors: Claudio Orlandi and Peter Scholl

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Abstract

A cryptographic accumulator is an algorithm to compute a short representation of a set, and it is possible to compute a witness for each element in the accumulated set which is used for the membership test. It has been considered as an interesting primitive for various applications.

Laconic cryptography is an emerging field within cryptography which enables realizing cryptographic tasks with asymptotically optimal communication in just two messages. And recently, several laconic private set intersection (PSI) protocols have been proposed from accumulators. In such protocols, the receiver has a potentially large input, and with accumulator schemes, the size of their protocol message is independent of the receiver's input size. The size of the second message by the sender is linear in the sender's input set size. There have been two constructions from RSA and bilinear pairing, but no one is quantum secure.

In this work, we aim to construct a laconic PSI protocol from lattice-based accumulators. We first recall RSA-based and pairing-based accumulators which have been used to construct laconic PSI protocols. We then move to the current two lattice-based accumulator constructions, one with trapdoor and the other in Merkle-tree style without trapdoor, and discuss their essential security properties. Meanwhile, we look into the possibilities to transform a lattice-based incremental hash function into an accumulator. Finally, we try to build a laconic or succinct PSI protocol from lattice-based accumulators and incremental hash functions.

Resumé

En kryptografisk akkumulator er en algoritme til beregning af en kort repræsentation af et sæt, og det er muligt at beregne et vidne for hvert element i det akkumulerede sæt, som bruges til medlemskabstesten. Den er blevet betragtet som en interessant primitiv metode til forskellige anvendelser.

Lakonisk kryptografi er et nyt område inden for kryptografi, som gør det muligt at gennemføre kryptografiske opgaver med asymptotisk optimal kommunikation i kun to meddelelser. For nylig er der blevet foreslået adskillige lakoniske protokoller med Private Set Intersection (PSI) fra akkumulatorer. I disse protokoller har modtageren et potentielt stort input, og med akkumulatorordninger er størrelsen af deres protokolmeddelese uafhængig af modtagerens inputstørrelse. Størrelsen af den anden meddelelse fra afsenderen er lineær i forhold til afsenderens størrelse af inputmængden. Der har været to konstruktioner ud fra RSA og bilineær parring, men ingen af dem er kvantsikre.

Dette speciale har til hensigt at konstruere en lakonisk PSI-protokol fra gitterbaserede akkumulatorer. Indledningsvis gennemgås RSA-baserede og parringsbaserede akkumulatorer, som er blevet brugt til at konstruere lakoniske PSI-protokoller. Herefter behandles de to nuværende lattice-baserede akkumulatorkonstruktioner, den ene med falddør og den anden Merkle-træ uden falddør, og diskuterer deres væsentlige sikkerhedsegenskaber. Samtidig undersøges mulighederne for at omdanne en lattice-baseret inkrementel hashfunktion til en akkumulator. Afslutningsvis forsøger specialet at opbygge en lakonisk eller kortfattet PSI-protokol ud fra lattice-baserede akkumulatorer og inkrementelle hash-funktioner.

Acknowledgments

Read, try to design and prove, behave as an adversary and a simulator, . . . – At Aarhus, I have gained fundamental knowledge in the field of cryptography. Now, I dare to say that I am a cryptography student.

I would like to express my appreciation to my thesis and project advisors, *Claudio Orlandi*, *Peter Scholl* and *Mark Simkin*. I cannot complete my projects and thesis without their help. I will never forget the days that Mark taught me everything I needed for the projects step by step, Claudio demonstrated how to turn an idea into a real research project, and Peter gave me inspiration every time when I shared with him what I had learned.

It is those great people that have convinced me what a wise decision I made two years ago to come to Aarhus for my master's study. I never expected that the two years can be so fast and now I have to say goodbye to them. Thanks to *Nico Döttling*, who has offered me a PhD position in Saarbrücken, I can continue with the exciting adventure in cryptography. We will definitely meet at some point later in the research world.

I have been meeting so many artists in my life, like pianists, film makers, painters and architects. Sometimes I admire their life since they are always presenting attractive works to the world. But on second thought, why can't I? Cryptography is indeed an art as well. I wish one day I could be proud to say that I am a cryptographer – both a scientist and an artist.

Chuanwei Lin 林傳威*,*
Aarhus, June 2022.

Contents

Chapter 1

Introduction

Cryptographic accumulators were proposed by Benaloh and de Mare as a decentralized alternative to digital signatures [Bd94]. Since their introduction, accumulators have been considered as interesting primitives for various applications, e.g., time-stamping and membership testing mechanisms [Bd94], fail-stop signatures [BP97], anonymous credential systems and group signatures [Nyb96, CL02, Ngu05, CKS09], ring signatures [DKNS04], redactable signatures [PS14], sanitizable signatures [CJ10], homomorphic signatures [ABC+12], digital cash [AWSM07, CG10, MGGR13], etc.

Basically, an accumulator scheme is an algorithm to compute a short representation of a set, and it is possible to compute a *witness* for each element in the accumulated set which is used for the membership test. The accumulators have several characteristics – robustness [Bd94], dynamic [CL02, ATSM09], universality [LLX07, ATSM09] and compactness [BP97, AWSM07, JS15]. In this work, we focus on the *one-shot* accumulator [BCY20], i.e., we do not stress dynamic changes to the accumulated set, and we only consider membership proofs.

The most common types of accumulators are RSA-based [Bd94, CL02, LLX07] and pairing-based [Ngu05, CKS09, DT08, ATSM09], depending on the underlying security assumptions. Alternative constructions are based on Paillier's trapdoor permutation [WWP08] and vector commitments [CF13]. However, the security of those schemes could theoretically be defeated using Shor's algorithm on a quantum computer [Sho94].

Recently, there have been two quantum-secure constructions from lattices, one compact construction with trapdoor using basis delegation technique [JS15] and the other in Merkle-tree style without trapdoor [LLNW16]. But both accumulators don't have many applications since they require trapdoor or tree construction, which are not practical in multi-party settings. It raises the following question:

Is it possible to construct a compact accumulator from lattices without trapdoor?

It seems possible, and incremental hash functions [BGG94, CDv+03, LKMW19] seem to be a plausible candidate to construct an accumulator. Indeed, an accumulator scheme is a hash function with membership test. In the case of hashing with incremental hash functions, if one element of the set is modified, the new

hash value can be derived by updating the old one instead of re-computing the hash from scratch – If we could make it, the new accumulator could be dynamic. We attempt to transform a lattice-based incremental hash function to an accumulator, which raises another question:

What is the key to transform a hash function to an accumulator?

The answer is obvious – security guarantee for the membership test. The Merkle-tree construction from lattices is collision-resistant from the hardness of *short integer solution*. However, with a lattice-based incremental hash function, during our various attempts, we haven't found any solution to prevent an adversary to fabricate a witness for an element not in the accumulated set since we cannot have a proper hardness assumption.

As for the applications of accumulators, in this work, we focus on private set intersection (PSI) protocols. A PSI protocol allows two parties holding private sets to jointly compute the intersection without revealing any other information about their input sets to each other. There have been numerous works [Mea86, FNP04, KS05, DCW13, PSSZ15, KKRT16, PRTY19, PRTY20] on efficient PSI protocols. In many of the applications, there is a powerful server with a large database who communicates with weak clients with small databases. However, most PSI protocols incur a significant computational overhead on both parties.

Laconic cryptography [CDG+17, QWW18] is an emerging field within cryptography which enables realizing cryptographic tasks with asymptotically optimal communication in just two messages. Laconic PSI protocols aim at minimizing the overhead for the party holding the small set and accumulator schemes help to map a large set into a small representation. A *succinct* but not laconic PSI protocol was constructed from the RSA-based accumulator with trapdoor [ADT11]. Two laconic PSI protocol from trapdoor-less accumulators have been proposed recently, one from RSA-based accumulators [ABD+21] and the other from pairing-based accumulators [ALOS22]. Still, no construction is quantum-secure. We try to answer the following question:

Can we construct PSI from current lattice-based accumulators?

Unfortunately, the current lattice-based accumulators cannot fit into the laconic PSI paradigm. Due to the basis delegation technique and tree construction, they are more complex than the ideal accumulators for PSI. We show our attempt to build a laconic PSI protocol from incremental hashing, where we still need to explore how to make it secure.

Even though in our work we have not got any satisfactory solutions to the problems above, we don't think the work ends here – To find a compact, lattice-based, trapdoor-less accumulator is not only to complete the types of accumulators, but to help to construct efficient protocols in the large data scale against quantum adversary as well.

1.1 Organization

In this work, we will demonstrate some of the current accumulator constructions under different security assumptions, their essential properties and their applications to build a PSI protocol. Moreover, we will point out the obstacles to solve the problems mentioned above.

Specifically, in Chapter 2, we provide some background of lattices and cryptographic primitives. Then, in Chapter 3, we introduce zero-knowledge proofs and Stern's protocol from lattices.

In Chapter 4, after presenting the definitions and properties of cryptographic accumulator schemes, we show two constructions from RSA and pairing respectively. Then we show how they are used to build a laconic PSI protocol.

Besides those two, in Chapter 5, we introduce two flavours of lattice-based accumulators, one compact but with trapdoor, the other in Merkle-tree style but without trapdoor. We show the Stern's type zero-knowledge proof system for accumulated values at the end of this chapter.

Then, in Chapter 6, we present lattice-based incremental hash functions in detail, which seem a candidate to build an accumulator. Finally, in Chapter 7, we show our attempts and failures in the following aspects:

- apply Stern's protocol for accumulated values for the lattice-based accumulator with trapdoor;
- transform a lattice-based incremental hash function into an accumulator;
- construct a PSI protocol from the current lattice-based accumulators;
- construct a laconic PSI protocol from a lattice-based incremental hash function.

Chapter 2

Preliminaries

In this chapter we will give an introduction to lattices and the necessary cryptographic background used in the remaining chapters. In Section 2.1, we give an overview of the notation in this work. In Section 2.2, we introduce the basic definitions of lattices and the different concepts used in lattice-based cryptography. In Section 2.3, we present some basic cryptographic primitives.

2.1 Notation

Matrices, vectors and sets Matrices are denoted by bold face capital letters, and vectors are denoted by bold face lower-case letters. For a vector $\boldsymbol{v}$, we denote by $\|\boldsymbol{v}\|$ the length of $\boldsymbol{v}$ in the ℓ_2 norm. Given a set $X = \{x_1, \ldots, x_m\}$, where $x_i \in \mathbb{Z}_q$ we write X_{-i} for the set $X_{-i} := X \setminus \{x_i\}$. We write $P(X, s)$ for the polynomial $\prod_{x \in X}(x - s)$ of degree $|X|$ in $\mathbb{Z}_q$ whose roots are all the elements in X and we write $P(X, s) = \sum_{i=0}^{|X|} p(X, i)s^i$ for its coefficients. Let $\mathsf{B}(m, n)$ be the set of all vectors in $\{0, 1\}^m$ that have Hamming weight n. Let $\mathcal{S}_m$ be the set of all permutations of m elements.

Assignments and choices If S is a set, then $U(S)$ denotes the uniform distribution on S, and we denote sampling x uniformly at random from S either with $x \leftarrow\!\$\, U(S)$ or $x \leftarrow\!\$\, S$. If χ is a probability distribution over a set S, then $x \leftarrow\!\$\, \chi$ denotes sampling $x \in S$ according to χ. If χ is a distribution over a set S, then $\boldsymbol{X} \leftarrow\!\$\, \chi(S^{(n \times m)})$ denotes generating an $n \times m$ matrix $\boldsymbol{X}$ by sampling each of its entries independently from S according to χ.

Functions and algorithms A function $\mu(\cdot)$ is *negligible* in n if for every positive polynomial $p(\cdot)$, $\mu(n) \leq 1/p(n)$ holds for any large enough n and we denote it as $\mathsf{negl}(n)$. Similarly, by $\mathsf{poly}(n)$ we denote any function $g = O(n^c)$ for some constant c. Let PPT denote probabilistic polynomial-time. If $\mathcal{A}$ is a PPT algorithm, $y \leftarrow\!\$\, \mathcal{A}(x)$ denotes running $\mathcal{A}$ on input x with randomly chosen coins and assigning the output to y.

Indistinguishability Given two probability ensembles U and V, we say that

- U and V are perfectly indistinguishable, written $U \triangleq V$, if $U_x = V_x$ for every x.

- U and V are statistically indistinguishable, written $U \overset{\text{s}}{\approx} V$, if the statistical distance between U and V, i.e., $\sum_x |U(x) - V(x)|$, is negligible in the length of x.
- U and V are computationally indistinguishable, written $U \overset{\text{c}}{\approx} V$, if for any PPT algorithm D, $|\Pr[D(x) = 1 \mid x \leftarrow\!\$\, U] - \Pr[D(x) = 1 \mid x \leftarrow\!\$\, V]|$ is negligible in the length of x.

Orthogonalization For any set $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k\} \subset \mathbb{R}^m$ of linearly independent vectors, let $\tilde{\boldsymbol{S}} = \{\tilde{\boldsymbol{s}}_1, \ldots, \tilde{\boldsymbol{s}}_k\}$ denote its *Gram-Schmidt orthogonalization*, defined as follows: $\tilde{\boldsymbol{s}}_1 = \boldsymbol{s}_1$, and for $i \in [2, k]$, the vector $\tilde{\boldsymbol{s}}_i$ is the component of $\boldsymbol{s}_i$ orthogonal to the linear span of $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{i-1})$.

2.2 Lattices

In this section, we recall some basic definitions and computational problems, which are obtained from several resources [MR09, Pei16].

2.2.1 Basic Definitions

Definition 2.1 (Lattice)**.** Let $\mathbb{R}^m$ be the m-dimensional Euclidean space. An m-dimensional lattice $\Lambda \in \mathbb{R}^m$ is a set that is both

- *an additive subgroup*: $\mathbf{0} \in \Lambda$, and $\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{y} \in \Lambda$ for every $\boldsymbol{x}, \boldsymbol{y} \in \Lambda$;
- *discrete*: every $\boldsymbol{x} \in \Lambda$ has a neighborhood in $\mathbb{R}^m$ in which $\boldsymbol{x}$ is the only lattice point.

Alternatively, lattices can also be characterized with reference to some basis vectors.

Definition 2.2 (Bases)**.** An m-dimensional lattice $\Lambda \in \mathbb{R}^m$ is a set

$$\Lambda = \left\{ \sum_{i=1}^{k} c_i \boldsymbol{b}_i \mid c_i \in \mathbb{Z} \text{ and } \boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \in \mathbb{R}^m \right\}$$

of all integral combination of k linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \in \mathbb{R}^m$ ($m \geq k$). The integers k and m are called the *rank* and *dimension* of the lattice, respectively. The matrix $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k] \in \mathbb{R}^{m \times k}$, having the basis vectors as columns, is called a *lattice basis*.

Using the matrix form of basis vectors, a lattice can be represented in a more compact form as $\Lambda = \Lambda(\boldsymbol{B}) = \{\boldsymbol{B}\boldsymbol{c} \mid \boldsymbol{c} \in \mathbb{Z}^k\}$. A lattice is called *integer lattice* if $\Lambda \subseteq \mathbb{Z}^m$. For the rest of this work, we focus on *full-rank* lattices, where $k = m$. In particular, we restrict our attention to q-ary lattices, a special family of full-rank integer lattices.

Definition 2.3 (q-ary lattices)**.** A q-ary lattice is specified by a parity check matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ for some positive integer n and positive integer modulus q. The associated full rank m-dimensional lattice is defined as

$$\Lambda^{\perp}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{x} = \mathbf{0} \bmod q\}$$

.

Definition 2.4 (Primitive matrix)**.** A matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ is *primitive* if its columns generate all of $\mathbb{Z}_q^n$, i.e., $\boldsymbol{A} \cdot \mathbb{Z}^m (\text{mod } q) = \mathbb{Z}_q^n$.

We assume a uniform $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ is primitive since for any fixed constant $C > 1$ and any $m \geq Cn \log q$, a uniformly random $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ is primitive, except with $2^{-\Omega(n)} = \mathsf{negl}(n)$ probability.

2.2.2 Computational Problems

The *minimum distance* $\lambda_1(\Lambda)$ of a lattice Λ is the Euclidean length, i.e., ℓ_2 norm. More generally, the ith successive minimum $\lambda_i(\Lambda)$ is the smallest radius r such that Λ contains i linearly independent vectors of norm at most r.

There are several computational problems on lattices that have direct importance to cryptography. Two of the standard worst-case approximation problems are: Approximate Shortest Vector Problem (SVP_γ), Decisional Approximate SVP (GapSVP_γ) and Approximate Shortest Independent Vectors Problem (SIVP_γ). Both problems are parameterized by an *approximation factor* $\gamma = \gamma(m)$ as a function of the lattice dimension m.

Definition 2.5 (SVP_γ)**.** Given a basis $\boldsymbol{B}$ of a full-rank m-dimensional lattice, find a nonzero lattice vector $\boldsymbol{Bx}$ (with $\boldsymbol{x} \in \mathbb{Z}^m \setminus \{\boldsymbol{0}\}$) such that $\|\boldsymbol{Bx}\| \leq \gamma \cdot \|\boldsymbol{By}\|$ for any $\boldsymbol{y} \in \mathbb{Z}^m \setminus \{\boldsymbol{0}\}$.

Definition 2.6 (GapSVP_γ)**.** Given a basis $\boldsymbol{B}$ of a full-rank m-dimensional lattice where either $\lambda_1(\Lambda) \leq 1$ or $\lambda_1(\Lambda) > \gamma(n)$, determine which is the case.

Definition 2.7 (SIVP_γ)**.** Given a basis $\boldsymbol{B}$ of a full-rank m-dimensional lattice, find a set of m linearly independent lattice vectors $\boldsymbol{Bx}_1, \ldots, \boldsymbol{Bx}_m \in \Lambda(\boldsymbol{B})$ such that $\|\boldsymbol{Bx}_i\| \leq \gamma \cdot \lambda_m(\Lambda(\boldsymbol{B}))$ for all i.

2.2.3 Hardness Assumptions

There are two main average-case problems that underlie most lattice-based cryptographic schemes – Short Integer Solution (SIS) and Learning with Error (LWE).

In a short integer solution (SIS) problem [Ajt96, MR04], we are given a set of uniformly random elements of a certain large finite additive group, we aim to find a sufficiently "short" nontrivial integer combination of them that sums to zero. Formally,

Definition 2.8 (SIS in the ℓ_2 norm)**.** Given an integer q, a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and a real β, find an integer vector $\boldsymbol{e} \in \mathbb{Z}^m$ such that $\boldsymbol{Ae} = \boldsymbol{0} \bmod q$ and $\|\boldsymbol{e}\| \leq \beta$.

Remark. Without the constraint on $\|\boldsymbol{e}\|$, the problem can be solved via Gaussian elimination. Similarly, it must satisfy that $\beta < q$ because otherwise $\boldsymbol{e} = (q, 0, \ldots, 0) \in \mathbb{Z}^m$ would always be a trivial solution.

A sequence of works has been proposed about the hardness of the SIS problem relative to worst-case lattice problems [Ajt96, MR04, GPV08, MP13].

Theorem 2.1 ([GPV08])**.** *For any poly-bounded m, any $\beta = \mathsf{poly}(n)$ and for any prime $q \geq \beta \cdot \omega(\sqrt{n \log n})$, the average-case $\mathsf{SIS}_{n,q,m,\beta}$ is as hard as SIVP_γ in the worst-case for some $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$.*

There are two approaches that can solve SIS: lattice reduction algorithms [SE94, GN08, CN11] and combinatorial algorithms [SS79, CP91, Wag02, BGLS19]. The lattice reduction algorithm aims to find a short basis for an input lattice, and hence, a solution to shortest vector problem. But when $m \gg n$, some combinatorial algorithms are more practical [BGLS19].

The learning with error (LWE) problem and the SIS problem are syntactically similar.

Definition 2.9 (LWE)**.** Let n, q be positive integers, and let χ be a distribution over $\mathbb{Z}_q$. For an $\boldsymbol{s} \in \mathbb{Z}_q^n$, the LWE distribution $A_{\boldsymbol{s},\chi}$ is the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random and an integer error $e \in \mathbb{Z}_q$ from χ, and outputting the pair $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. There are two main kinds of computational LWE problems:

- *search problem*: recover the secret $\boldsymbol{s} \in \mathbb{Z}_q^n$ given m samples drawn from the LWE distribution $A_{\boldsymbol{s},\chi}$.
- *decision problem*: distinguish m samples drawn from the LWE distribution $A_{\boldsymbol{s},\chi}$ from uniformly random samples.

A sequence of works has been proposed about the hardness of the LWE problem relative to worst-case lattice problems [Reg05, Pei09, LM09, BLP+13].

Theorem 2.2 ([Reg05])**.** *For any poly-bounded m, any modulus $q \leq 2^{\mathsf{poly}(n)}$ and any (discretized) Gaussian error distribution χ of parameter $\alpha q \geq 2\sqrt{n}$ where $0 < \alpha < 1$, the decisional $\mathsf{LWE}_{n,q,\chi,m}$ problem is as hard as GapSVP_γ and SIVP_γ in the worst case for some $\gamma = \tilde{O}(n/\alpha)$.*

2.2.4 Discrete Gaussian Distribution

Many works from lattices in cryptography reply on Gaussian distributions over lattices, called *discrete Gaussians*.

Definition 2.10 (Gaussian function)**.** For any $s > 0$, which is taken to be $s = 1$ when omitted, the Gaussian function $\rho_{s,c} : \mathbb{R}^m \to \mathbb{R}$ centered at $\boldsymbol{c} \in \mathbb{R}^m$ of parameter s is defined as

$$\rho_{s,\boldsymbol{c}}(\boldsymbol{x}) = e^{-\frac{\pi \|\boldsymbol{x}-\boldsymbol{c}\|^2}{s^2}}$$

.

Definition 2.11 (Discrete Gaussian distribution)**.** For any $\boldsymbol{c} \in \mathbb{R}^m$, $s > 0$ and m-dimensional lattice Λ, the discrete Gaussian distribution $D_{\Lambda,s,\boldsymbol{c}}$ over Λ with center $\boldsymbol{c}$ and parameter s is defined as

$$D_{\Lambda,s,\boldsymbol{c}}(\boldsymbol{x}) = \frac{\rho_{s,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{s,\boldsymbol{c}}(\boldsymbol{\Lambda})}$$

.

The ℓ_2 norm of vectors sampled from the discrete Gaussian distribution is small with non-negligible probability [MR04], presented by Lemma 2.1.

Lemma 2.1. *Let $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ be a primitive matrix, and s be a Gaussian parameter with $s \geq \omega\sqrt{\log m}$. For an m-dimensional full-rank lattice $\Lambda^{\perp}(\boldsymbol{A})$, and $\boldsymbol{c} \in \mathbb{R}^m$,*

$$\Pr_{\boldsymbol{x} \leftarrow D_{\Lambda^{\perp}(\boldsymbol{A}), s, \boldsymbol{c}}} \left[\|\boldsymbol{x} - \boldsymbol{c}\| > s\sqrt{m} \right] \leq \mathsf{negl}(m)$$

.

For a vector $\boldsymbol{e}$ chosen from an appropriate discrete Gaussian distribution over $\mathbb{Z}^m$, the vector $\boldsymbol{A}\boldsymbol{e} \bmod q$ is close to a uniformly chosen vector from $\mathbb{Z}_q^n$ [GPV08], presented by Lemma 2.2.

Lemma 2.2. *Let $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ be a primitive matrix. For any $s \geq \omega\sqrt{\log m}$, the distribution of $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e} \bmod q \in \mathbb{Z}_q^n$ is statistically close to uniform over $\mathbb{Z}_q^n$, where $\boldsymbol{e}$ is chosen from $D_{\mathbb{Z}^m, s, \boldsymbol{0}}$.*

$\underline{\mathsf{SampleD}\,(\boldsymbol{B}, s, \boldsymbol{c})}$

Input:

- a basis $\boldsymbol{B}$ of a lattice $\Lambda \subseteq \mathbb{R}^m$,
- a positive real parameter $s \leq \left\|\tilde{\boldsymbol{B}}\right\| \cdot \omega\sqrt{\log m}$,
- a center vector $\boldsymbol{c} \in \mathbb{R}^n$.

Output:

- a random lattice vector $\boldsymbol{x} \in \Lambda$ drawn from a distribution statistically close to $D_{\Lambda, s, \boldsymbol{c}}$.

Figure 2.1: Gaussian sampling algorithm

Given a basis $\boldsymbol{B}$ for a lattice Λ, one can efficiently sample points in Λ with discrete Gaussian distribution for some large s [GPV08], presented by Theorem 2.3 and Figure 2.1.

Theorem 2.3. *Given a basis $\boldsymbol{B}$ of an m-dimensional lattice Λ, a parameter $s \geq \left\|\tilde{\boldsymbol{B}}\right\| \cdot \omega\sqrt{\log m}$, and a center $\boldsymbol{c} \in \mathbb{R}^m$, there is a PPT algorithm that outputs a sample from a distribution that is statistically close to $D_{\Lambda, s, \boldsymbol{c}}$.*

2.2.5 Basis Delegation

There exists a PPT algorithm that extend a lattice to an arbitrary higher-dimensional extension, without any loss of quality [CHKP12], presented by Lemma 2.3 and Figure 2.2.

Lemma 2.3. *Given a basis $\boldsymbol{B}$ of a m-dimensional lattice $\Lambda^{\perp}(\boldsymbol{A})$ for some primitive $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and an arbitrary matrix $\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$, there exists a* PPT *algorithm that outputs a basis $\boldsymbol{B'}$ of $\Lambda^{\perp}(\boldsymbol{A'} = \boldsymbol{A}||\bar{\boldsymbol{A}})$ such that $\|\boldsymbol{B'}\| = \|\boldsymbol{B}\|$. Moreover, the statement holds even if the columns of $\boldsymbol{A'}$ are permuted arbitrarily.*

$\underline{\mathsf{BasisDel}\left(\boldsymbol{T}_{\boldsymbol{A}}, \boldsymbol{A}, \bar{\boldsymbol{A}}\right)}$

Input:

- an arbitrary primitive matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$
- an arbitrary basis $\boldsymbol{T}_{\boldsymbol{A}}$ of $\Lambda^{\perp}(\boldsymbol{A})$,
- an arbitrary matrix $\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$.

Output:

- a basis $\boldsymbol{T}_{\boldsymbol{A'}}$ of $\Lambda^{\perp}(\boldsymbol{A'}) \subseteq \mathbb{Z}^{m+\bar{m}}$ where $\boldsymbol{A'} = \boldsymbol{A}||\bar{\boldsymbol{A}}$ such that $\left\|\tilde{\boldsymbol{T}}_{\boldsymbol{A'}}\right\| = \left\|\tilde{\boldsymbol{T}}_{\boldsymbol{A}}\right\|$

Figure 2.2: Basis delegation algorithm

2.3 Cryptographic Background

Definition 2.12 (Pseudorandom generator)**.** Let $G : \{0,1\}^* \mapsto \{0,1\}^*$ and $\ell(\cdot)$ be a polynomial such that for any input $s \in \{0,1\}^{\lambda}$ we have $G(s) \in \{0,1\}^{\ell(\lambda)}$. Then, G is a secure pseudorandom generator if the following two conditions hold

- *expansion*: $\ell(\lambda) > \lambda$
- *pseudorandomness*: the two distributions

$$\left\{G(s) \mid s \leftarrow \{0,1\}^{\lambda}\right\} \text{ and } \left\{r \mid r \leftarrow \{0,1\}^{\ell(\lambda)}\right\}$$

  are computationally indistinguishable.

Definition 2.13 (Collision-resistant hash functions)**.** A hash function $h : \{0,1\}^* \mapsto \{0,1\}^{\ell(\lambda)}$, generated by $\mathcal{H}$ which on input a security parameter $k \in \{0,1\}^{\kappa(\lambda)}$, is collision-resistant if for any PPT algorithm $\mathcal{A}$, it holds that

$$\Pr\left[\left(x \neq x'\right) \wedge \left(h(x) = h(x')\right) \;\middle|\; (x, x') \leftarrow \mathcal{A}(h)\right] \leq \mathsf{negl}(\lambda).$$

Definition 2.14 (Commitment scheme)**.** A commitment scheme is a triplet of algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver})$ such that

- Gen is a PPT algorithm that, on input security parameter 1^{λ}, outputs some public parameters pp containing a definition of the message space, the randomness space and the commitment space.

- Com is a deterministic polynomial-time algorithm that, on input pp, a message x and some randomness ρ, outputs a commitment c.
- Ver is a deterministic polynomial-time algorithm that, on input pp, a message x, a commitment c and some randomness ρ, outputs a bit $b \in \{0,1\}$.

A commitment scheme has three essential properties – correctness, hiding and binding.

Definition 2.15 (Correctness property)**.** A commitment scheme is correct, if for any message x and any randomness ρ,

$$\Pr[\mathsf{Ver}(x,c,\rho)=1 \mid c \leftarrow \mathsf{Com}(x;\rho)] = 1.$$

Definition 2.16 (Hiding property)**.** A commitment scheme is said computationally (resp. statistically, resp. perfectly) hiding if, for any two messages x_0 and x_1, the two distributions

$$\{c \mid c \leftarrow \mathsf{Com}(x_0;\rho), \rho \leftarrow \$\} \text{ and } \{c \mid c \leftarrow \mathsf{Com}(x_1;\rho), \rho \leftarrow \$\}$$

are computationally (resp. statistically, resp. perfectly) indistinguishable.

Definition 2.17 (Binding property)**.** A commitment scheme is binding if for every algorithm $\mathcal{A}$, it holds that

$$\Pr\left[\begin{array}{c|c} (x \neq x') \wedge (\mathsf{Ver}(x,c,\rho)=1) \wedge \\ (\mathsf{Ver}(x',c,\rho)=1) \end{array} \middle| (x,x',\rho,\rho',c) \leftarrow \mathcal{A}(\mathsf{pp})\right] \leq \mathsf{negl}(\lambda).$$

If we restrict $\mathcal{A}$ to being PPT, then the scheme is computationally binding. If the computation time of $\mathcal{A}$ is unbounded, then the scheme is statistically binding.

Definition 2.18 (Random oracle)**.** A random oracle is a black box that responds to every unique query with a uniformly random response. If a query is repeated, it responds the same way every time that query is submitted.

Chapter 3

Zero-knowledge Proofs from Lattices

Zero-knowledge proofs are an important tool for many cryptographic protocols and applications. Such proofs enable a prover to prove a statement by interacting with a verifier without revealing anything more than the statement itself. In Section 3.1, we introduce the definitions and properties of zero-knowledge proofs. In Section 3.2, we show the Stern's type protocol from lattices. In Section **??**, we show the Schnorr's type protocol from lattices.

3.1 Zero-Knowledge Proofs

3.1.1 Interactive Proof Protocols

A proof is a protocol in which a *prover* outputs a complete proof to a *verifier* who decides on the validity. The interaction allows the prover and the verifier to interact with each other. We recall some basic definitions in this section [BG93, Gol01, Dam02].

Definition 3.1 (Interactive protocol)**.** A two-party interactive protocol is a triplet $(\mathsf{Init}, \mathcal{A}, \mathcal{B})$ where Init is an initialization algorithm produces a pair $(\mathsf{in}_{\mathcal{A}}, \mathsf{in}_{\mathcal{B}})$ on input 1^{λ}, and where $\mathcal{A}$ and $\mathcal{B}$ are two stateful algorithms, called *parties*. The parties receive their inputs $\mathsf{in}_{\mathcal{A}}$ and $\mathsf{in}_{\mathcal{B}}$ respectively, exchange messages and finally one party, say $\mathcal{A}$, produces the output of the protocol.

An execution of the protocol is a sequence:

$$
\begin{aligned}
&\mathsf{state}_{\mathcal{A}} \leftarrow \mathcal{A}(\mathsf{in}_{\mathcal{A}})\\
&\mathsf{state}_{\mathcal{B}} \leftarrow \mathcal{B}(\mathsf{in}_{\mathcal{B}})\\
&(\mathsf{msg}_{\mathcal{A}}[0], \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}})\\
&\vdots\\
&(\mathsf{msg}_{\mathcal{B}}[i], \mathsf{state}_{\mathcal{B}}) \leftarrow \mathcal{B}(\mathsf{state}_{\mathcal{B}}, \mathsf{msg}_{\mathcal{A}}[i-1])\\
&(\mathsf{msg}_{\mathcal{A}}[i], \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, \mathsf{msg}_{\mathcal{B}}[i])\\
&\vdots\\
&\mathsf{out} \leftarrow \mathcal{B}(\mathsf{state}_{\mathcal{B}}, \mathsf{msg}_{\mathcal{A}}[n])
\end{aligned}
$$

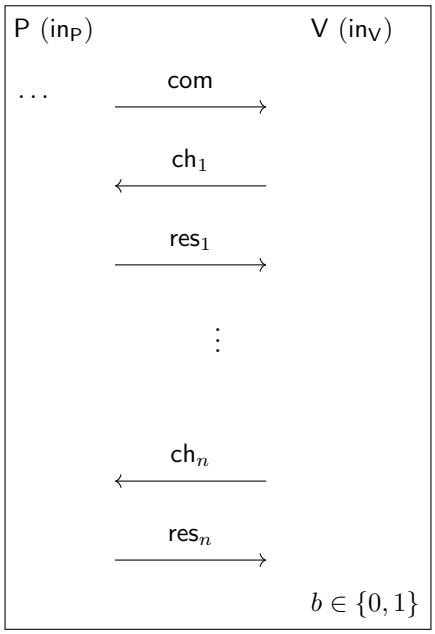

Figure 3.1: Structure of a m-round interactive proof with $m = 2n + 1$

The sequence of exchanged messages is called the *transcript* of the execution, denoted as $\mathsf{view}\left(\langle \mathcal{A}\left(\mathsf{in}_{\mathcal{A}}\right), \mathcal{B}\left(\mathsf{in}_{\mathcal{B}}\right)\rangle\right)$.

In the rest of this work, the algorithm Init and the state of both parties are implicit. An algorithm has rewindable black-box access to a party $\mathcal{A}$ if the algorithm can copy the state of $\mathcal{A}$ at any moment, relaunch $\mathcal{A}$ from a previously copied state and query $\mathcal{A}$ on input messages.

An *interactive proof* is a special kind of two-party interactive protocol which involves a *prover* P and a *verifier* V, and P tries to prove a statement to V, illustrated in Figure 3.1. The first message sent by P is a commitment com. From this commitment V produces a first challenge ch_1, and V answers with a response res_1, followed by a next challenge ch_2 from V, and so on. After receiving the last response from P, V outputs a bit. The interactive proof is m-round with $m = 2n + 1$.

3.1.2 Proofs of Knowledge

Let R be a binary relation, i.e., R is a subset of $\{0,1\}^* \times \{0,1\}^*$ where if $(x, w) \in R$ then the length of w is at most $\mathsf{poly}(|x|)$. For $(x, w) \in R$, we call w a *witness* for x. Define L_R to be the set of x's for which there exist w such that $(x, w) \in R$. A proof of knowledge is defined as follows:

Definition 3.2 (Proof of knowledge)**.** Let κ: $\{0,1\}^* \to [0 \ldots 1]$ be a function. A protocol (P, V) is a proof of knowledge for a relation R, with knowledge error κ, if the following properties are satisfied:

- *completeness*: if P and V are honest and on input x and private input w to P where $(x, w) \in R$, then V always accepts.

- *knowledge soundness*: There exists a knowledge extractor Ext where on input x and rewindable black-box access to the prover, it attempts to output w such that $(x, w) \in R$. For every prover P^* and $x \in L_R$, Ext satisfies the following condition. Let $\varepsilon(x)$ be the probability that V accepts on input x after interacting with P^*. If $\varepsilon(x) > \kappa(x)$, then upon input x and oracle access to P^*, the extractor Ext returns $w : (x, w) \in R$ within an expected number of steps bounded by

$$\frac{|x|^c}{\varepsilon(x) - \kappa(x)}$$

.

A proof of knowledge has soundness error ε if a prover P^* without knowledge of the witness cannot convince the verifier V with probability greater than ε assuming that the underlying problem is hard. If a prover P^* can succeed with a probability greater than ε, then the existence of Ext implies that P^* can be used to compute a valid witness.

Definition 3.3 (Special soundness)**.** A $(2\mu + 1)$-round interactive protocol is called $(k_1, \ldots, k_\mu)$-special sound, if there exists an efficient algorithm that computes a witness from any set of $K := \prod_{i=1}^{\mu} k_i$ accepting transcripts.

To prove that a protocol is a proof of knowledge for a relation R, it suffices to demonstrate that the protocol has the special soundness property [AC20, Lemma 3].

The spirit of the definition of zero-knowledge is that whatever the verifier learns, he could have learned by himself without any interaction with the prover. Formally,

Definition 3.4 (Zero knowledge)**.** An interactive proof is computationally (resp. statistically, resp. perfectly) zero-knowledge if there exists a PPT algorithm (simulator) Sim whose output distribution is computationally (resp. statistically, resp. perfectly) indistinguishable from the distribution $\mathsf{view}(\langle \mathsf{P}(x, w), \mathsf{V}(x) \rangle)$ obtained with a (cheating) V^*.

Assume that the verifier is honest, we do not consider the possibility that a malicious verifier can extract extra information from the prover.

Definition 3.5 (Honest-verifier zero knowledge)**.** An interactive proof is computationally (resp. statistically, resp. perfectly) honest-verifier zero-knowledge (HVZK) if there exists a PPT algorithm (simulator) Sim whose output distribution is computationally (resp. statistically, resp. perfectly) indistinguishable from the distribution $\mathsf{view}(\langle \mathsf{P}(x, w), \mathsf{V}(x) \rangle)$ obtained with an honest V.

3.1.3 Σ-protocol

An interactive proof which has 3-round structure (commitment, challenge and response) as shown in Figure 3.1 is called a *sigma protocol* [Dam02]. The Greek letter Σ visualizes the structure of the protocol.

Definition 3.6 (Σ-protocol)**.** Let R be a binary relation, $R = (x, w)$, where x is a public input to both P and V, and w is private input to P. Consider the following protocol $\mathcal{P}$:

1. P sends a message a.
2. V sends a random t-bit string challenge e.
3. P sends a response z, and V decides whether to accept or reject based on x, a, e, z.

A protocol $\mathcal{P}$ is said to be a Σ-Protocol for a relation R if it satisfies:

- *completeness*: The 3-round protocol $\mathcal{P}$ is complete: if P, V follow the protocol on input x and private input w to P where $(x, w) \in R$, the verifier V always accepts.
- *special soundness:* From (a, e, z) and (a, e', z') such that $(x, a, e, z) \neq (x, a, e', z')$ but both accepted by the verifier V, where $e \neq e'$, then one can efficiently compute a w such that $(x, w) \in R$.
- *special honest-verifier zero-knowledge*: There exists a simulator Sim, where Sim takes as input (x, e) and returns the conversation (a, e, z) distributed exactly as a real conversation by $(\mathsf{P}, \mathsf{V})(x)$, where e is the challenge.

The special soundness property implies that a Σ-protocol for R is always an interactive proof system for L_R with error probability 2^{-t}. And the soundness error can be reduced by parallel repetition.

Lemma 3.1. *The parallel repetition ℓ times of a Σ-protocol for R with challenge length t yields a new Σ-protocol for R with challenge length $\ell \cdot t$. The soundness error is reduced to $2^{-\ell \cdot t}$.*

The challenge can be set to any arbitrary length.

Lemma 3.2. *If there exists a Σ-protocol for R with challenge length t, then there exists a Σ-protocol for R with challenge length t', for any t'.*

Note that a Σ-protocol is only zero-knowledge if the verifier is honest but a Σ-protocol protocol can be converted to one with non-honest verifier zero-knowledge [HL10, Chapter 6.5]. A Σ-protocol can also be made non-interactive using the Fiat-Shamir construction [FS87].

3.2 Stern's Protocol

3.2.1 System Construction

Stern proposed a zero-knowledge protocol to prove the knowledge of a syndrome decoding solution [Ste94b]. The protocol achieves a soundness error of 2/3 and thus a malicious prover can cheat the verifier with probability of 2/3. By repeating the protocol τ times, the soundness error can be reduced to $(2/3)^{\tau}$.

Init: On input 1^λ, it generates a random matrix $\mathbb{Z}_q^{n\times m}$, chooses a random vector $\boldsymbol{x} \in \mathsf{B}(m, m/2)$ and computes $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$. It outputs $(\mathsf{pk}, \mathsf{sk}) = (\boldsymbol{y}, \boldsymbol{x})$.
P, V: The common inputs are $\boldsymbol{A}$ and $\boldsymbol{y}$. The prover P's auxiliary input is $\boldsymbol{x}$. They interact as follows (the random tape and random strings for the commitment scheme are implicit):

- **Round 1** (com): The prover P choose a random permutation π over $[m]$ and a random vector $\boldsymbol{r} \in \mathbb{Z}_q^m$ and send commitments $\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3$ computed as
  - $\mathsf{com}_1 = \mathsf{Com}(\pi, \boldsymbol{A}\boldsymbol{r})$,
  - $\mathsf{com}_2 = \mathsf{Com}(\pi(\boldsymbol{r}))$,
  - $\mathsf{com}_3 = \mathsf{Com}(\pi(\boldsymbol{x} + \boldsymbol{r}))$.
- **Round 2** (ch): The verifier V sends a random challenge $\mathsf{ch} \in \{1, 2, 3\}$ to the prover P.
- **Round 3** (res): The prover P responses upon the challenge ch received:
  - If $\mathsf{ch} = 1$, reveal com_2 and com_3. P sends $\boldsymbol{s} = \pi(\boldsymbol{x})$ and $\boldsymbol{t} = \pi(\boldsymbol{r})$.
  - If $\mathsf{ch} = 2$, reveal com_1 and com_3. P sends $\phi = \pi$ and $\boldsymbol{u} = \boldsymbol{x} + \boldsymbol{r}$.
  - If $\mathsf{ch} = 3$, reveal com_1 and com_2. P sends $\psi = \pi$ and $\boldsymbol{v} = \boldsymbol{r}$.
- **Output**: The verifier V checks upon ch it chose as follow:
  - If $\mathsf{ch} = 1$, check that $\mathsf{com}_2 = \mathsf{Com}(\boldsymbol{t})$, $\mathsf{com}_3 = \mathsf{Com}(\boldsymbol{s} + \boldsymbol{t})$ and $\boldsymbol{s} \in \mathsf{B}(m, m/2)$.
  - If $\mathsf{ch} = 2$, check that $\mathsf{com}_1 = \mathsf{Com}(\phi, \boldsymbol{A}\boldsymbol{u} - \boldsymbol{y})$ and $\mathsf{com}_3 = \mathsf{Com}(\phi(\boldsymbol{u}))$.
  - If $\mathsf{ch} = 3$, check that $\mathsf{com}_1 = \mathsf{Com}(\psi, \boldsymbol{A}\boldsymbol{v})$ and $\mathsf{com}_2 = \mathsf{Com}(\psi(\boldsymbol{v}))$.

  The verifier V outputs 1 if all the checks are passed, otherwise outputs 0.

Figure 3.2: Stern's type protocol from lattices

Many works have been proposed to optimize and implement the protocol [Vér97, GG07, MGS11, ACBH13, GPS22, FJR21, FJR22a].

Stern's protocol can be modified as a lattice-based one [KTX08], shown in Figure 3.2. The protocol has perfect completeness and at most 2/3 soundness error.

It is a proof system for the relation

$$R = \{(\boldsymbol{A}, \boldsymbol{y}); \boldsymbol{x} : \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q\}.$$

3.2.2 Zero-knowledge Proof

Now we show that the proof system is zero-knowledge.

Theorem 3.1. *The protocol in Figure 3.2 is statistically zero knowledge when the commitment scheme* Com *is statistically-hiding and computationally-binding.*

Proof. We construct a simulator Sim on input $\boldsymbol{A}$ and $\boldsymbol{y}$ and given oracle access to a (cheating) verifier V^* outputting a simulated transcript, which is statistically indistinguishable from the distribution $\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{x}), \mathsf{V}^*(\boldsymbol{A}, \boldsymbol{y})\rangle)$.

The simulator Sim chooses a random value $\mathsf{ch}^* \in \{1, 2, 3\}$ for the challenge that the cheating verifier V^* will *not* choose. The random tape is implicit.

Case $\mathsf{ch}^* = 1$: The simulator computes $\boldsymbol{x}' \in \mathbb{Z}_q^m$ such that $\boldsymbol{A}\boldsymbol{x}' = \boldsymbol{y}$. It samples a random permutation π' over $[m]$, a random vector $\boldsymbol{r}' \in \mathbb{Z}_q^m$ and random strings for the commitment scheme $\rho_1', \rho_2', \rho_3'$. Then, it sends the commitments to V^* computed as follows

- $\mathsf{com}_1' = \mathsf{Com}(\pi', \boldsymbol{A}\boldsymbol{r}'; \rho_1')$,
- $\mathsf{com}_2' = \mathsf{Com}(\pi'(\boldsymbol{r}'); \rho_2')$,
- $\mathsf{com}_3' = \mathsf{Com}(\pi'(\boldsymbol{x}' + \boldsymbol{r}'); \rho_3')$.

Upon a challenge ch from V^*, the simulator Sim outputs a transcript as follows:

- If $\mathsf{ch} = 1$, Sim outputs $\perp$ and halts (since it predicts that V^* will not choose $\mathsf{ch} = 1$).
- If $\mathsf{ch} = 2$, Sim outputs $((\mathsf{com}_1', \mathsf{com}_2', \mathsf{com}_3'), 2, (\pi', \boldsymbol{x}' + \boldsymbol{r}', \rho_1', \rho_3'))$. In this case, the view of the real protocol is

  $$\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{x}), \mathsf{V}^*(\boldsymbol{A}, \boldsymbol{y})\rangle) = ((\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3), 2, (\pi, \boldsymbol{x}+\boldsymbol{r}, \rho_1, \rho_3)).$$

  Assume that $(\pi', \boldsymbol{r}', \rho_1', \rho_3') = (\pi', \boldsymbol{r} + \boldsymbol{x} - \boldsymbol{x}', \rho_1, \rho_3)$, then $\mathsf{com}_1' = \mathsf{com}_1, \mathsf{com}_3' = \mathsf{com}_3$, and the responses are identical in both distributions. Since the commitment scheme is statistically hiding, the distributions of com_2' and com_2 are statistically indistinguishable. This concludes that the simulator's output and the view in the real protocol are statistically indistinguishable.
- If $\mathsf{ch} = 3$, Sim outputs $((\mathsf{com}_1', \mathsf{com}_2', \mathsf{com}_3'), 3, (\pi', \boldsymbol{r}', \rho_1', \rho_2'))$. In this case, the view of the real protocol is

  $$\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{x}), \mathsf{V}^*(\boldsymbol{A}, \boldsymbol{y})\rangle) = ((\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3), 3, (\pi, \boldsymbol{r}, \rho_1, \rho_2)).$$

  Assume that $(\pi', \boldsymbol{r}', \rho_1', \rho_2') = (\pi, \boldsymbol{r}, \rho_1, \rho_2)$ and from the statistical hiding property of the commitment scheme, we can conclude that the two distributions are statistically indistinguishable.

Case $\mathsf{ch}^* = 2$: The simulator samples a random permutation π' over $[m]$, a random vector $\boldsymbol{r}' \in \mathbb{Z}_q^m$ and another $\boldsymbol{x}' \in \mathsf{B}(m, m/2)$ and random strings for the commitment scheme $\rho_1', \rho_2', \rho_3'$. Then, it sends the commitments to V^* computed as follows

- $\mathsf{com}'_1 = \mathsf{Com}(\pi', \boldsymbol{A}\boldsymbol{r}'; \rho'_1)$,
- $\mathsf{com}'_2 = \mathsf{Com}(\pi'(\boldsymbol{r}'); \rho'_2)$,
- $\mathsf{com}'_3 = \mathsf{Com}(\pi'(\boldsymbol{x}' + \boldsymbol{r}'); \rho'_3)$.

Upon a challenge ch from V^*, the simulator Sim outputs a transcript as follows:

- If $\mathsf{ch} = 1$, Sim outputs $((\mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3), 1, (\pi'(\boldsymbol{x}'), \pi'(\boldsymbol{r}'), \rho'_2, \rho'_3))$. In this case, the view of the real protocol is

  $$\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{x}), \mathsf{V}^*(\boldsymbol{A}, \boldsymbol{y})\rangle) = ((\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3), 1, (\pi(\boldsymbol{x}), \pi(\boldsymbol{r}), \rho_2, \rho_3)).$$

  Let χ be a permutation over $[m]$ such that $\chi(\boldsymbol{x}') = \boldsymbol{x}$. Assume that $(\pi', \boldsymbol{r}', \rho'_2, \rho'_3) = (\pi \circ \chi^{-1}, \chi(\boldsymbol{r}), \rho_2, \rho_3)$, then $\pi(\boldsymbol{x} = \pi'(\boldsymbol{x}'))$, $\pi(\boldsymbol{r}) = \pi'(\boldsymbol{r}')$, $\mathsf{com}'_1 = \mathsf{com}_1$ and $\mathsf{com}'_3 = \mathsf{com}_3$, and the responses are identical in both distributions. Since the commitment scheme is statistically hiding, the distributions of com'_1 and com_1 are statistically indistinguishable. This concludes that the simulator's output and the view in the real protocol are statistically indistinguishable.
- If $\mathsf{ch} = 2$, $\perp$ and halts (since it predicts that V^* will not choose $\mathsf{ch} = 2$).
- If $\mathsf{ch} = 3$, Sim outputs $((\mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3), 3, (\pi', \boldsymbol{r}', \rho'_1, \rho'_2))$. In this case, the view of the real protocol is

  $$\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{x}), \mathsf{V}^*(\boldsymbol{A}, \boldsymbol{y})\rangle) = ((\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3), 3, (\pi, \boldsymbol{r}, \rho_1, \rho_2)).$$

  Same as that in the case $\mathsf{ch} = 3$ under case $\mathsf{ch}^* = 1$, we can conclude that the two distributions are statistically indistinguishable.

Case $\mathsf{ch}^* = 3$: The simulator samples a random permutation π' over $[m]$, a random vector $\boldsymbol{r}' \in \mathbb{Z}_q^m$ and another $\boldsymbol{x}' \in \mathsf{B}(m, m/2)$ and random strings for the commitment scheme $\rho'_1, \rho'_2, \rho'_3$. Then, it sends the commitments to V^* computed as follows

- $\mathsf{com}'_1 = \mathsf{Com}(\pi', \boldsymbol{A}(\boldsymbol{x}' + \boldsymbol{r}') - \boldsymbol{y}; \rho'_1)$,
- $\mathsf{com}'_2 = \mathsf{Com}(\pi'(\boldsymbol{r}'); \rho'_2)$,
- $\mathsf{com}'_3 = \mathsf{Com}(\pi'(\boldsymbol{x}' + \boldsymbol{r}'); \rho'_3)$.

Upon a challenge ch from V^*, the simulator Sim outputs a transcript as follows:

- If $\mathsf{ch} = 1$, Sim outputs $((\mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3), 1, (\pi'(\boldsymbol{x}'), \pi'(\boldsymbol{r}'), \rho'_2, \rho'_3))$. In this case, the view of the real protocol is

  $$\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{x}), \mathsf{V}^*(\boldsymbol{A}, \boldsymbol{y})\rangle) = ((\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3), 1, (\pi(\boldsymbol{x}), \pi(\boldsymbol{r}), \rho_2, \rho_3)).$$

  Same as that in the case $\mathsf{ch} = 1$ under case $\mathsf{ch}^* = 2$, with a help of a permutation χ over $[m]$, we can conclude that the two distributions are statistically indistinguishable.

– If $\mathsf{ch} = 2$, Sim outputs $((\mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3), 2, (\pi', \boldsymbol{x}' + \boldsymbol{r}', \rho'_1, \rho'_3))$. In this case, the view of the real protocol is

$$\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{x}), \mathsf{V}^*(\boldsymbol{A}, \boldsymbol{y})\rangle) = ((\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3), 2, (\pi, \boldsymbol{x}+\boldsymbol{r}, \rho_1, \rho_3)).$$

Same as that in the case $\mathsf{ch} = 2$ under case $\mathsf{ch}^* = 1$, we can conclude that the two distributions are statistically indistinguishable.

– If $\mathsf{ch} = 3$, Sim outputs $\perp$ and halts (since it predicts that V^* will not choose $\mathsf{ch} = 3$).

By the above arguments, the distribution of the simulator's output conditioned on it is not $\perp$, is statistically indistinguishable from the view in the real protocol. □

3.2.3 Proof of Knowledge

Now we show that the proof system is 3-special sound, and thus it is a proof of knowledge for a relation $R = \{(\boldsymbol{A}, \boldsymbol{y}); \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q\}$.

Theorem 3.2. *If the commitment scheme* Com *is statistically hiding and computationally binding, then there exists an efficient knowledge extract* Ext *which on input* 3 *valid responses* $(\mathsf{res}_1, \mathsf{res}_2, \mathsf{res}_3)$ *to the same commitment series* $(\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$ *outputs a* $\boldsymbol{x}^*$ *such that* $((\boldsymbol{A}, \boldsymbol{y}); \boldsymbol{x}^*) \in R$.

Proof. Given the 3 valid responses to $(\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$,

$$\begin{cases} \mathsf{res}_1 = (\boldsymbol{s}, \boldsymbol{t}) \\ \mathsf{res}_2 = (\phi, \boldsymbol{u}) \\ \mathsf{res}_3 = (\psi, \boldsymbol{v}) \end{cases}$$

The validity of res_1 implies that $\boldsymbol{s} \in \mathsf{B}(m, m/2)$

Since the 3 responses are all valid, from the binding property of the commitment scheme, it holds that

$$\begin{cases} \mathsf{com}_1 = \mathsf{Com}(\phi, \boldsymbol{A}\boldsymbol{u} - \boldsymbol{y} \bmod q) = \mathsf{Com}(\psi, \boldsymbol{A}\boldsymbol{r} \bmod q) \\ \mathsf{com}_2 = \mathsf{Com}(\boldsymbol{t}) = \mathsf{Com}(\psi(\boldsymbol{v})) \\ \mathsf{com}_3 = \mathsf{Com}(\boldsymbol{s} + \boldsymbol{t}) = \mathsf{Com}(\phi(\boldsymbol{u})) \end{cases}$$

and $\phi = \psi$, $\boldsymbol{A}\boldsymbol{u} - \boldsymbol{y} = \boldsymbol{A}\boldsymbol{r} \bmod q$, $\boldsymbol{t} = \psi(\boldsymbol{v})$ and $\boldsymbol{s} + \boldsymbol{t} = \phi(\boldsymbol{u})$.

The extractor Ext proceeds as follows. It sets $\boldsymbol{x}' = \psi^{-1}(\boldsymbol{s})$.

Since $\boldsymbol{s} \in \mathsf{B}(m, m/2)$, we have $\boldsymbol{x}' \in \mathsf{B}(m, m/2)$. Moreover, note that $\psi(\boldsymbol{x}') + \psi(\boldsymbol{v}) = \phi(\boldsymbol{u})$, which implies that $\boldsymbol{x}' + \boldsymbol{v} = \boldsymbol{u}$.

Now we have $\boldsymbol{A}\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{u} - \boldsymbol{A}\boldsymbol{v} = \boldsymbol{y} \bmod q$, which means that $\boldsymbol{x}'$ is a valid witness. □

Chapter 4

Cryptographic Accumulators

Cryptographic accumulators, originally introduced in [Bd94], allow the compact representation of an arbitrarily large set of elements S that support proofs of membership in the underlying set. Accumulators are an essential primitive for various applications where a compact representation of a set is needed and element membership should be provable. In Section 4.1, we introduce the definition and properties for one-shot accumulators. In Section 4.2, we show the paradigm to construct a secure private set intersection from an accumulator. In Section 4.3 and Section 4.4, we introduce the RSA-based and pairing-based accumulators respectively.

4.1 One-Shot Accumulators

A cryptographic accumulator [Bd94] is defined as a compact representation of a set S that supports proofs of membership in the underlying set. Given a membership witness w for an element x accumulated, and the accumulated value v of the accumulator, it can be efficiently verified that the element x is a valid member of the accumulated set. In practice, an accumulator can be *dynamic* that it supports both element addition (*additive*) and deletion (*subtractive*) [CL02]. Moreover, an accumulator is called *positive* or *negative* if it only supports membership or non-membership proofs respectively, and *universal* if it supports both membership and non-membership proofs [ATSM09]. In this work, we focus on a simplified, one-shot accumulator [BCY20] where we only consider *static* accumulator, i.e., we do not stress dynamic changes to the accumulated set, and we only consider membership proofs. Formally,

Definition 4.1 (One-shot cryptographic accumulator)**.** An one-shot accumulator scheme parameterized by a domain $\mathcal{D}$ consists of four PPT algorithms $(\mathsf{Setup}, \mathsf{Acc}, \mathsf{Wit}, \mathsf{Ver})$ defined as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: given the security parameter λ, it sets up the global public parameters pp for the accumulator system.
- $\mathsf{Acc}(\mathsf{pp}, S) \to v_S$: given the public parameters pp and a set $S \subseteq \mathcal{D}$, it returns an accumulated value v_S.

- $\mathsf{Wit}(\mathsf{pp}, S, x) \to w$: given the public parameters pp, a set $S \subseteq \mathcal{D}$ and an element $x \in S$, it returns a membership witness w for the element x.
- $\mathsf{Ver}(\mathsf{pp}, v_S, x, w) \to \mathtt{accept}/\mathtt{reject}$: given the public parameters pp, an accumulated value v_S, an element x and a witness w, it checks whether w proves that x is a member of the set S.

Remark. In the definition shown above, we need *a trusted party* to process the Setup phase. While in a trapdoor-based setting [CL02, BCD+17], a trapdoor sk is generated in Setup phase as well. An accumulator manager processes the trapdoor sk and generates witnesses with sk.

We now present the security properties for an one-shot accumulator. An one-shot accumulator must be correct and collision-resistant.

An one-shot accumulator is correct if an honestly produced witness w corresponding to x in a set S can always be accepted by verifying the membership of x in an accumulator v_S. Formally,

Definition 4.2 (Correctness)**.** For all security parameters λ, all values x in a set S to be accumulated:

$$Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ v_S \leftarrow \mathsf{Acc}(\mathsf{pp}, S); \\ w \leftarrow \mathsf{Wit}(\mathsf{pp}, S, x); \\ \mathsf{Ver}(\mathsf{pp}, v_S, x, w) = \mathtt{accept} \end{array}\right] = 1$$

An one-shot accumulator is collision-resistant (sound) if it is computationally infeasible to construct a membership witness for an element not in the accumulated set. Formally,

Definition 4.3 (Collision-resistance)**.** For any PPT adversary $\mathcal{A}$, there exists a negligible function negl in the security parameter λ such that:

$$Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ v_S \leftarrow \mathsf{Acc}(\mathsf{pp}, S); \\ (x, w) \leftarrow \mathcal{A}^{\mathsf{Wit}}(v_S); \\ x \notin S : \mathsf{Ver}(\mathsf{pp}, v_S, x, w) = \mathtt{accept} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

The most common types of accumulators are RSA-based [Bd94, CL02, LLX07] and pairing-based [Ngu05, CKS09, DT08, ATSM09], depending on the underlying security assumptions. There have been some constructions from lattice [JS15, LLNW16]. We will show them later.

4.2 Private Set Intersection from Accumulators

One application of the one-shot accumulator we focus in this work is to construct a private set intersection protocol. In a two-party private set intersection (PSI) problem, Alice and Bob hold sets A and B respectively and would like to jointly compute the intersection $C = A \cap B$, without revealing any information about the elements not in the intersection. In a laconic private set intersection (ℓ-PSI) problem, we consider one party (the 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 the other party (the client) with a string y can send the server a short message allowing it to learn y if $y \in S$ and nothing otherwise [CDG+17, ABD+21, ALOS22].

A cryptographic accumulator is a proper candidate to represent the set of the server. We consider the following accumulator:

- The accumulation function $A = \mathsf{Acc}(X)$ allows to compress a set X into a small representation A.
- The witness w for an element y is not a function of y, instead, it is generated from the set of X except the element x, i.e., $w = \mathsf{Wit}(X_{-y})$.
- The verification function can be described as that if $\mathsf{Ver}(\mathsf{Acc}(X), x, w) = 1$, there exist two functions ψ, ϕ such that $\phi\left(\psi(x), X_{-y}\right) = \mathsf{Acc}(X)$ and the functions output elements in a group where it holds that $\phi\left(\psi(y)^{\beta}, X\right) = \phi(\psi(y), X)^{\beta}$ for all y, X and scalar β.

In a two-party PSI protocol, the receiver holds a large set X and acts as the accumulator - it accumulates the set, creates the witnesses for each element in the set, and performs the Ver algorithm with the value sent by the sender, i.e., the sender sends $\psi(y)$ for elements in his set Y and then the receiver finishes the Ver algorithm with the function ϕ. The paradigm is shown in Figure 4.1. For simplicity, in the figure, we consider private set membership where the sender has a single element as input. This can be extended into a PSI protocol by sending a permuted series of (T_i, U_i) for each y_i in the sender's set Y.

Sender $\mathcal{S}$		**Receiver** $\mathcal{R}$
Input: a string y		Input: a set X
sample random β		sample random α
	$R = \mathsf{Acc}(X)^{\alpha} \longrightarrow$	
	$\longleftarrow T = R^{\beta}, U = \psi(y)^{\beta}$	
		$\forall x_i \in X:$ let $X_{-i} = X \setminus \{x_i\}$
		If $\phi(U, X_{-x})^{\alpha} = T$, output $y = x_i$

Figure 4.1: Private set membership (PSM) test constructed from a cryptographic accumulator

In this paradigm, the receiver sends a randomized accumulated value $R = \mathsf{Acc}(X)^{\alpha}$ for some random α. The sender replies with $\psi(y)^{\beta}$ and R^{β}. Finally, the receiver computes witnesses for each subset of X of size $|X| - 1$ and checks whether the element owned by the sender matches the one removed from the set X. Security of the protocols derived from this paradigm follows from the usage of the randomizers α, β and the fact that the receiver cannot perform a brute force attack on y since it is computationally infeasible to generate a witness for an element not in the accumulated set.

4.3 RSA-based Accumulator

4.3.1 Accumulator Construction from RSA

We show the construction and security analysis of an one-shot RSA-based accumulator [Bd94], and show how it has been used in constructing a laconic private set intersection protocol [ADT11, ABD+21]. The construction is shown in Figure 4.2.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$:
  1. Sample $N \leftarrow\!\!\$\, \mathsf{RSA}(\lambda)$ such that $N = PQ$ where P and Q are safe prime numbers.
  2. Pick a random generator $g \in \mathbb{Z}_N^*$.
  3. Sample distinct primes $p_1, \ldots, p_\ell$ respectively for a polynomially-sized universe $\mathcal{U} = \{1, \ldots, \ell\}$.
  4. Return $\mathsf{pp} = \{N, g, \{p_i\}_{i\in[\ell]}\}$.
- $\mathsf{Accumulate}(\mathsf{pp}, X) \to v_X$:
  Return $v_X = g^{\prod_{i\in X} p_i} \mod N$ where p_i is the pairwise prime of x_i.
- $\mathsf{WitCreate}(\mathsf{pp}, X, x_i) \to w$:
  Return $w = g^{\prod_{j\in X_{-i}} p_j} \mod N$ where p_j is the pairwise prime of x_j.
- $\mathsf{Verify}(\mathsf{pp}, v_X, x, w) \to$ `accept`/`reject`:
  If $v_X = w^p \bmod N$ where p is the prime corresponding to x, then return `accept`, otherwise, return `reject`.

Figure 4.2: RSA-based Accumulator

Correctness property is straightforward. For the set $X = \{x_1, \ldots, x_m\}$ to be accumulated and $\{p_1, \ldots, p_m\}$ to be the pairwise primes, let $v_X = g^{\prod_{i\in X} p_i} \bmod N$ be the accumulated value and $w = g^{\prod_{j\in X_{-i}} p_j} \mod N$ be a witness for an element $x_i \in X$ produced honestly. Since $w^{p_i} = g^{p_i \cdot \prod_{j\in X_{-i}} p_j} = v_X \bmod N$, an honestly produced witness will always be accepted.

Before analyzing the collision-resistance property, we introduce the strong RSA assumption [BP97].

Assumption 4.1 (Strong RSA)**.** Any PPT algorithm $\mathcal{A}$, given a randomly chosen RSA modulus N, a random $z \in \mathbb{Z}_n$, has negligible probability to output a pair (y, e) such that $y^e = z \mod N$, e is a prime and $e < n$.

Theorem 4.1. *The accumulator construction of Figure 4.2 is collision-resistant under the strong RSA assumption.*

Proof. Let $\mathcal{A}$ be a PPT adversary can break collision-resistance property of the

one-shot accumulator, we show a construction of a PPT algorithm $\mathcal{B}$ that can break the strong RSA assumption.

$\mathcal{B}$ is given (N, z) where N is a valid RSA modulus and $z \in_R \mathbb{Z}_n$. We show that $\mathcal{B}$ can compute (y, e) such that $y^e = z \mod N$ with non-negligible probability.

$\mathcal{A}$ breaks collision-resistance property of the accumulator, he can output a set $Y = \{p_1, \ldots, p_n\} \subset \mathbb{Z}_N$, a value $p' \in \mathbb{Z}_N \setminus Y$ and a witness $w' \in \mathbb{Z}_N^*$ such that $w'^{p'} = x^{p_1 \cdots p_n} \mod N$.

Let $e := p'$ and $r := p_1 \cdots p_n$. Since $p_1, \ldots, p_n$ and p' are distinct primes, with the extended Euclidean algorithm, we can compute $a, b \in \mathbb{Z}$ such that $ar + bp' = 1$. Then let $y := w'^a x^b$, we have $y^e = w'^{ea} x^{eb} = x^{ra} x^{p'b} = x \mod N$. $\mathcal{B}$ returns p' as the solution to the strong RSA problem. □

4.3.2 Laconic PSI from RSA-based Accmulator

Based on the RSA-based accumulator, we show a simple, passively secure version of ℓPSI protocol from RSA-based accumulators [ABD+21] in the random oracle model.

Before stating the security of the protocol, we introduce the ϕ-hiding assumption.

Assumption 4.2 (ϕ-hiding)**.** Let $\mathsf{Prime}(\kappa)$ denote the set of prime numbers of bit-length κ. Let $\mathsf{RSA}(\lambda) = \{N : N = PQ \wedge P, Q \in \mathsf{Prime}(\lambda/2) \wedge \gcd(P-1, Q-1) = 2\}$ and $\mathsf{RSA}_e(\lambda) = \{N : e | \phi(N)\}$ for any $e \leq 2^\lambda$. Any PPT algorithm $\mathcal{A}$ has negligible probability to distinguish N sampled from $\mathsf{RSA}(\lambda)$ and that sampled from $\mathsf{RSA}_e(\lambda)$.

Theorem 4.2. *The laconic private set intersection protocol shown in Figure 4.3 is correct and secure in the semi-honest model.*

Thanks to the randomizers, strong RSA and ϕ-hiding assumptions, there is no possibility for both parties to launch brute force attacks. We refer to [ABD+21] for the full proof in a hybrid model.

4.4 Pairing-based Accumulator

Before giving the construction of a pairing-based accumulator, we introduce bilinear pairings.

Definition 4.4 (Bilinear pairing)**.** Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic additive groups with prime order p generated by g_1 and g_2, respectively, and $\mathbb{G}_T$ be a cyclic multiplicative group with the same order p. Suppose there is an isomorphism: $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing satisfying the following properties:

1. Bilinearity: $e\left(u^a, v^b\right) = e\left(u, v\right)^{ab}$ for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$
2. Non-degeneracy: $e\left(g_1, g_2\right) \neq 1$.
3. Computability: There is an efficient algorithm to compute $e\left(u, v\right)$ for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$

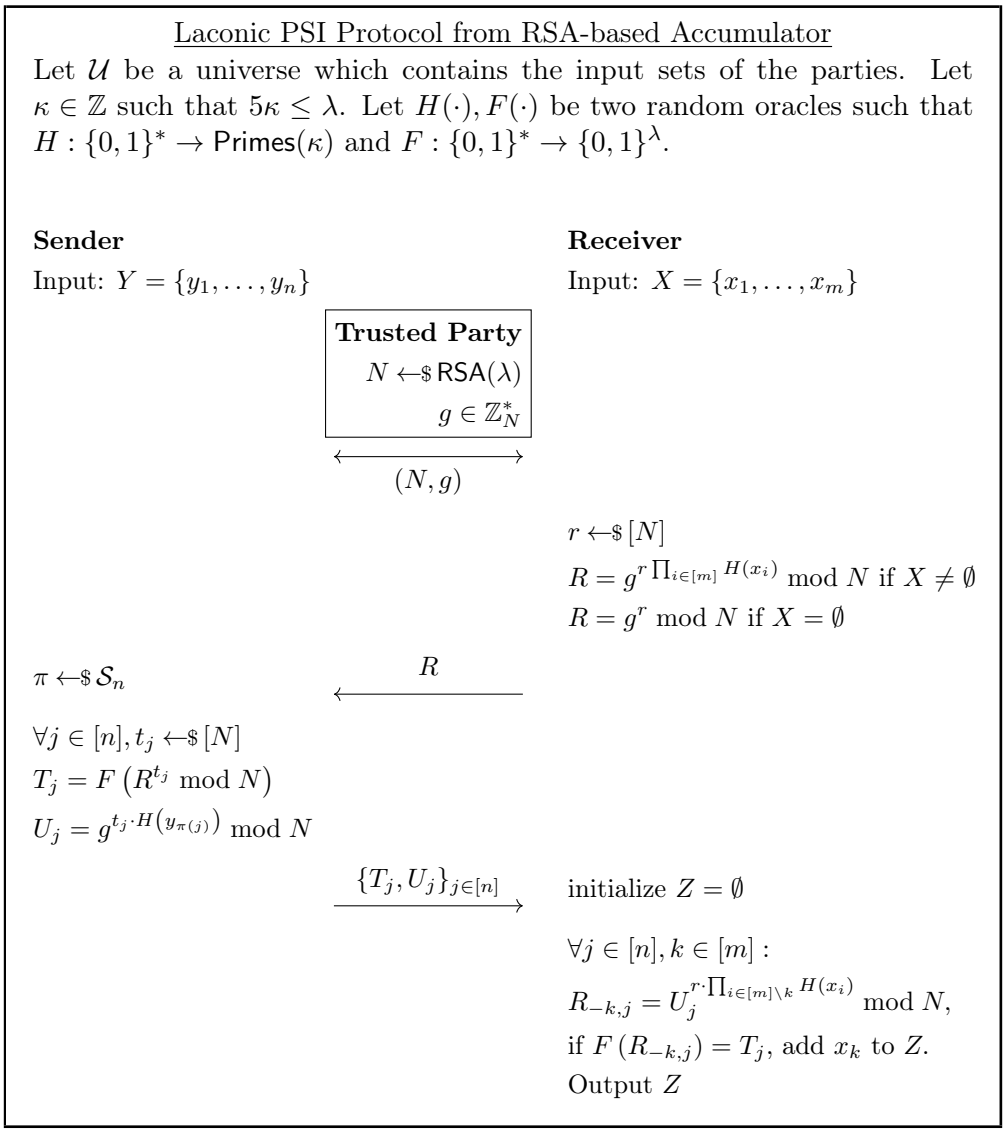

Figure 4.3: The laconic PSI protocols of [ABD+21] based on RSA-accumulator in the random oracle model

When $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is symmetric, called Type 1 pairing. When $\mathbb{G}_1 \neq \mathbb{G}_2$, the pairing is asymmetric. If there is an efficiently computable isomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, then the pairing is called Type 2 pairing. If there is no known isomorphism, then the pairing is called Type 3 pairing.

4.4.1 Accumulator Construction from Pairing

The construction of an accumulator from type-3 pairing is shown in 4.4 [Ngu05, KB21, ALOS22].

Remark (Multiplication of Polynomials in the Exponent). To evaluate a polynomial $p_k(x) = \sum_{i=0}^{k} c_i x^i$ of degree k at an unknown point a in the exponent

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$:
  1. Select a pairing instance $\mathsf{pi} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $|p| = \lambda$ and p is a prime.
  2. Pick a random $s \in \mathbb{Z}_p^*$, and let the accumulator's domain $\mathcal{D} = \mathbb{Z}_p - \{s\}$.
  3. Compute a $(q+3)$-tuple where q is the capacity of the accumulator $\left(g_1, g_1^s, g_1^{s^2}, \ldots, g_1^{s^q}, g_2, g_2^s\right)$.
  4. Return $\mathsf{pp} = \left(\mathsf{pi}, \left(g_1, g_1^s, g_1^{s^2}, \ldots, g_1^{s^q}, g_2, g_2^s\right)\right)$.
- $\mathsf{Acc}(\mathsf{pp}, S) \to v_S$:

  Return $v_S = g_1^{P(X,s)} = g_1^{\sum_{i=1}^{|X|}\left[p(X,i)\cdot s^i\right]} = \prod_{i=1}^{|X|}\left(g_1^{s^i}\right)^{p(X,i)}$.
- $\mathsf{Wit}(\mathsf{pp}, X, x_i) \to w$:

  Return $w = g_1^{P(X_{-i},s)} = g_1^{\sum_{i=1}^{|S|-1}\left[p(X_{-i},i)\cdot s^i\right]} = \prod_{i=1}^{|X|-1}\left(g_1^{s^i}\right)^{p(X_{-i},i)}$.
- $\mathsf{Ver}(\mathsf{pp}, v_S, x, w) \to$ `accept`/`reject`: If $e\left(v_X, g_2\right) = e\left(w, g_2^s \cdot g_2^x\right)$, then return `accept`, otherwise, return `reject`.

Figure 4.4: Pairing-based Accumulator

using k elements g^{a^i}, $i = \{0, \ldots, k\}$ as

$$g^{p_k(a)} = g^{\sum_{i=0}^{k} c_i a^i} = g^{c_0} \cdot (g^a)^{c_1} \cdot \left(g^{a^2}\right)^{c_2} \cdot \cdots \cdot \left(g^{a^k}\right)^{c_k}.$$

Now we show the correctness property. For the set $X = \{x_1, x_2, \ldots, x_m\}$ to be accumulated, let $v_X = \prod_{i=1}^{|S|}\left(g_1^{s^i}\right)^{p(X,i)}$ be the accumulated value and $w = \prod_{i=1}^{|X|-1}\left(g_1^{s^i}\right)^{p(X_{-i},i)}$ be a witness for an element $x_i \in X$ produced honestly. Note that

$$e(v_X, g_2) = e(g_1, g_2)^{\sum_{i=1}^{|X|}\left[p(X,i)\cdot s^i\right]}$$

$$e(w, g_2^s \cdot g_2^{x_i}) = e(g_1, g_2)^{(x_i+s)\cdot\sum_{i=1}^{|X|-1}\left[p(X_{-i},i)\cdot s^i\right]}$$

Since $\sum_{i=1}^{|X|}\left[p(X,i)\cdot s^i\right] = \prod_{x\in X}(x+s)$, we have $\sum_{i=1}^{|X|}\left[p(X,i)\cdot s^i\right] = (x_i + s)\cdot\sum_{i=1}^{|X|-1}\left[p(X_{-i},i)\cdot s^i\right]$, and $e\left(v_X, g_2\right) = e\left(w, g_2^s \cdot g_2^x\right)$ will be always accepted.

Before analyzing the collision-resistance property, we introduce the strong Diffie-Hellman assumption [Che06].

Assumption 4.3 (B-Strong Diffie-Hellman, B-SDH)**.** Let $\mathbb{G}_1 = \langle g_1\rangle$ and $\mathbb{G}_2 = \langle g_2\rangle$ be two cyclic group of prime order q and $s \in \mathbb{Z}_q^*$. Any PPT algorithm $\mathcal{A}$, given a $(B+3)$-tuple of elements $\left(g_1, g_1^s, g_1^{s^2}, \ldots, g_1^{s^B}, g_2, g_2^s\right) \in \mathbb{G}_1^{B+1} \times \mathbb{G}_2^2$, has negligible probability to output a pair $\left(x, g_1^{1/(x+s)}\right) \in \mathbb{Z}_q \times \mathbb{G}_1$.

Theorem 4.3. *The accumulator construction of Figure 4.4 is collision-resistant under the strong Diffie-Hellman assumption.*

Proof. Let $\mathcal{A}$ be a PPT adversary can break collision-resistance property of the one-shot accumulator, we show a construction of a PPT algorithm $\mathcal{B}$ that can break the B-SDH assumption.

$\mathcal{B}$ is given $\left(g_1, g_1^s, g_1^{s^2}, \ldots, g_1^{s^B}, g_2, g_2^s\right)$ where $s \in_R \mathbb{Z}_p^*$. We show that $\mathcal{B}$ can compute $\left(c, g_1^{1/(s+c)}\right)$ where $c \in \mathbb{Z}_p$ with non-negligible probability.

$\mathcal{A}$ breaks collision-resistance property of the accumulator, he can output a set $Y \subset \mathbb{Z}_p \setminus \{-s\}$, a value $y' \in \mathbb{Z}_p \setminus (\{-s\} \cup Y)$ and a witness $w \in \mathbb{G}_1$ such that $e\left(w, g_2^s \cdot g_2^y\right) = e\left(v_Y, g_2\right)$.

Note that $v_Y = g_1^{P(Y,s)}$, i.e. $v_Y = g_1^P(s)$. Since $y \notin Y$, there exists a polynomial $Q(s)$ of degree less than q such that $P(s) = Q(s)(y' + s) + d$ for a constant d. By the property of bilinear pairing, we have $e(w^{s+y'}, g_2) = e(g_1^{Q(s)(s+y')+d}, g_2)$, i.e. $w^{s+y'} = g_1^{Q(s)(s+y')+d}$. Then we can have $g_1^{1/(y'+s)} = \left[w \cdot g_1^{-Q(s)}\right]^{\frac{1}{d}}$. $\mathcal{B}$ returns $\left(y', \left[w \cdot g_1^{-Q(s)}\right]^{\frac{1}{d}}\right)$ as the solution to the B-SDH problem. □

4.4.2 Laconic PSI from Pairing-based Accumulator

Based on the pairing-based accumulator, we show a passively secure version of ℓPSI protocol from pairing-based accumulators [ALOS22] in the random oracle model.

The Game $\mathcal{G}^{\mathsf{SBDDH}}_{b,(B_1,B_2),\mathcal{A}}(\lambda)$

The game is parametrized by a stateful adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, a parameter (B_1, B_2) and a bit b.

1. $s \leftarrow\$ \mathbb{Z}_q^*$;
2. $y \leftarrow \mathcal{A}_1\left(\left\{g_1^{s^i}\right\}_{i=0}^{B_1}, \left\{g_2^{s^i}\right\}_{i=0}^{B_2}\right)$
3. $T_0 = e(g_1, g_2)^{1/(y+s)}$;
4. $T_1 \leftarrow\$ \mathbb{G}_T$;
5. $b' \leftarrow \mathcal{A}_2(T_b)$;
6. Output 1 if $b = b'$.

Figure 4.5: The (B_1, B_2)-Strong Bilinear Decisional Diffie-Hellman Game ((B_1, B_2)-SBDDH)

The protocol is correct from the correctness of the accumulator scheme. The security is under a natural assumption over pairing- friendly curves, called

(B_1, B_2)-Strong Bilinear Decisional Diffie-Hellman assumption, which is proved in a generic group model.

Assumption 4.4 ((B_1, B_2)-Strong Bilinear Decisional Diffie-Hellman Problem ((B_1, B_2)-SBDDH))**.** Consider the game $\mathcal{G}^{\text{SBDDH}}_{b,(B_1,B_2),\mathcal{A}}$ described in Figure 4.5. We say that the (B_1, B_2)-SBDDH problem is hard if, for every PPT adversary $\mathcal{A}$, the advantage

$$\text{Adv}^{\text{SBDDH}}_{(B_1,B_2),\mathcal{A}}(\lambda) := \left|\Pr\left[\mathcal{G}^{\text{SBDDH}}_{0,(B_1,B_2),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathcal{G}^{\text{SBDDH}}_{1,(B_1,B_2),\mathcal{A}}(\lambda) = 1\right]\right|$$

is negligible in the security parameter λ.

Theorem 4.4. *The laconic private set intersection protocol shown in Figure 4.6 is correct and secure in the semi-honest model.*

Security of the protocol follows from the use of randomizers and the fact that it is computationally infeasible for the receiver to perform a brute force attack due to the hardness of (B_1, B_2)-SBDDH. We refer to [ALOS22] for the full proof in a hybrid model.

Laconic PSI Protocol from Pairing-based Accumulator

Let $\mathcal{U}$ be a universe which contains the input sets of the parties. Let λ be security parameter and $H(\cdot)$ be a random oracles such that $H : \{0,1\}^* \to \{0,1\}^\lambda$. The parties agree on a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$ where the generators are $g_1, g_2, g_T = e(g_1, g_2)$ respectively.

Sender

Input: $Y = \{y_1, \ldots, y_n\}$

Trusted Party

$s \leftarrow\$ \mathbb{Z}_q^*, S' = g_1^s$

$S_i = g_2^{s^i}, i \in [0, m]$

$\xleftarrow{\qquad\qquad\qquad\qquad}$

$S' \qquad\qquad \{S_i\}_{i\in[m]}$

Receiver

Input: $X = \{x_1, \ldots, x_m\}$

$r \leftarrow\$ \mathbb{Z}_q$

$$R = \left(\prod_{i=0}^{m} S_i^{p(X,i)}\right)^r \text{ if } X \neq \emptyset$$

$R = (S_0)^r$ if $X = \emptyset$

$\pi \leftarrow\$ \mathcal{S}_n$

$\xleftarrow{\qquad R \qquad}$

$\forall j \in [n], t_j \leftarrow\$ \mathbb{Z}_q$

$T_j = H\left(e\left(g_1^{t_j}, R\right)\right)$

$U_j = \left(S' \cdot g_1^{-y_{\pi(j)}}\right)^{t_j}$

$\xrightarrow{\qquad \{T_j, U_j\}_{j\in[n]} \qquad}$

initialize $Z = \emptyset$

$\forall j \in [n], k \in [m]$:

$$R_{-k} = \left(\prod_{i=0}^{m-1} S_i^{p(X_{-k},i)}\right)^r, \text{ if}$$

$H\left(e\left(U_j, R_{-k}\right)\right) = T_j$, add x_k to Z.

Output Z

Figure 4.6: The laconic PSI protocols of [ALOS22] based on pairing-accumulator in the random oracle model

Chapter 5

Lattice-based Accumulators

There exist a few lattice-based accumulators, in trapdoor setting [JS15] and in trapdoorless setting [LLNW16, YAY+18]. Both constructions are secure based on the hardness of Short Integer Solution problem. In Section 5.1, we show the construction of a *compact* accumulator with trapdoor setting. In Section 5.2, we show the construction of a tree-style accumulator from lattice without trapdoor, and we show the method to apply Stern's protocol [Ste94a] to construct a zero-knowledge argument system.

5.1 Trapdoor Accumulator from Lattices

5.1.1 Accumulator Construction

The first compact accumulator scheme from lattices is proposed in a trapdoor-based style [JS15], shown in Figure 5.1.

At a high level, the messages are chosen from $\mathbb{Z}_q^{n\times m'}$ and the size of message space is Q. The Setup algorithm produces a statistically uniform matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ as pk along with a short basis $\boldsymbol{T_A}$ of lattice $\Lambda^{\perp}(\boldsymbol{A})$ as sk using the algorithm in Figure 2.1.

An accumulated value is the sum of the Q' messages in the set S,

$$v_S = \left[\sum_{i=1}^{Q'} \boldsymbol{B}_i\right] \in \mathbb{Z}_q^{n\times m'}$$

.

The witness is not a function of the element $\boldsymbol{B}_i$. Instead, it computes a matrix

$$\boldsymbol{F}_{\boldsymbol{B}_i} = \left[\boldsymbol{A} || \sum_{1\le j(\neq i)\le Q'} \boldsymbol{B}_j\right] \in \mathbb{Z}_q^{n\times(m+m')}$$

using the public key $\boldsymbol{A}$ and the set of X except the element $\boldsymbol{B}_i$. Then it samples a vector $w \in \Lambda^{\perp}(\boldsymbol{F}_{\boldsymbol{B}_i}) \subset \mathbb{Z}^{m+m'}$ as the witness following the distribution $D_{\Lambda^{\perp}(\boldsymbol{F}_{\boldsymbol{B}}),s,\boldsymbol{0}}$ using the secret key $\boldsymbol{T_A}$.

To check whether w is a valid witness for $\boldsymbol{B}$, it computes

$$\boldsymbol{F}_{\boldsymbol{B}} = [\boldsymbol{A}||v - \boldsymbol{B}] \in \mathbb{Z}_q^{n\times(m+m')}$$

Parameters: Let λ be the security parameter, integer $n = \Theta(\lambda)$, prime $q = \mathsf{poly}(\lambda)(n)$, dimension $m \geq 6n \lg q$, bound $L = O(\sqrt{m})$, Gaussian parameter $s \geq L \cdot \omega\left(\sqrt{\log(m+m')}\right)$ where $m' = \mathsf{poly}(\lambda)(\lambda) \in \mathbb{N}$, and message set $\mathcal{M} = \{\boldsymbol{B}_1, \ldots, \boldsymbol{B}_Q \in \mathbb{Z}_q^{n \times m'}\}$ where $Q = \mathsf{poly}(\lambda)(\lambda)$.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$:
  1. Generate a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and a basis $\boldsymbol{T}_{\boldsymbol{A}}$ for $\Lambda^\perp(\boldsymbol{A})$;
  2. Set the public key $\mathsf{pk} = \boldsymbol{A}$ and the secret key $\mathsf{sk} = \boldsymbol{T}_{\boldsymbol{A}}$.
- $\mathsf{Acc}(S) \to v_S$: Without loss of generality, suppose $X = \{\boldsymbol{B}_1, \ldots, \boldsymbol{B}_{Q'}\}$ for $X \subset \mathcal{U}$.
  Return $v_X = \left[\sum_{i=1}^{Q'} \boldsymbol{B}_i\right] \in \mathbb{Z}_q^{n \times m'}$.
- $\mathsf{Wit}(\mathsf{pk}, \mathsf{sk}, X, \boldsymbol{B}_i) \to w$:
  Return $w \leftarrow \mathsf{SampleD}\left(\mathsf{BasisDel}\left(\boldsymbol{T}_{\boldsymbol{A}}, \boldsymbol{A}, \sum_{1 \leq j (\neq i) \leq Q'} \boldsymbol{B}_j\right), s, \boldsymbol{0}\right)$.
- $\mathsf{Ver}(\mathsf{pk}, v_S, \boldsymbol{B}, w) \to$ `accept`/`reject`:
  1. Compute $\boldsymbol{F}_{\boldsymbol{B}} = [\boldsymbol{A} || (v_S - \boldsymbol{B})] \in \mathbb{Z}_q^{n \times (m+m')}$,
  2. Check if $\boldsymbol{F}_{\boldsymbol{B}} \cdot w = \boldsymbol{0} \mod q$ and $0 < ||w|| \leq s\sqrt{m+m'}$.
  3. If both checks pass, return `accept`, otherwise, return `reject`.

Figure 5.1: Trapdoor-based Accumulator from Lattice

and check if $w \in \Lambda^\perp(\boldsymbol{F}_{\boldsymbol{B}})$ and w is small.

We first show that the accumulator scheme is correct. For a set $X = \{\boldsymbol{B}_1, \ldots, \boldsymbol{B}_{Q'}\}$ to be accumulated, let $v_X = \sum_{i=1}^{Q'} \boldsymbol{B}_i$ and w be a witness for an element $\boldsymbol{B}_i \in X$ produced honestly. The witness is a short vector $\boldsymbol{d}_{\boldsymbol{B}_i}$ in the lattice $\Lambda^\perp(\boldsymbol{F}_{\boldsymbol{B}_i})$ where $\boldsymbol{F}_{\boldsymbol{B}_i} = \left[\boldsymbol{A} || \sum_{1 \leq j (\neq i) \leq Q'} \boldsymbol{B}_j\right]$ and $||\boldsymbol{d}_{\boldsymbol{B}_i}|| \leq s\sqrt{m+m'}$.

The witness creation algorithm has access to a short basis $\boldsymbol{T}_{\boldsymbol{A}}$ of the lattice $\Lambda^\perp(\boldsymbol{A})$. With $\boldsymbol{A}$, $\boldsymbol{T}_{\boldsymbol{A}}$ and $\sum_{1 \leq j (\neq i) \leq Q'} \boldsymbol{B}_j$ as input, the $\mathsf{BasisDel}$ Algorithm outputs a basis $\boldsymbol{T}_{\boldsymbol{F}_{\boldsymbol{B}_i}}$ of $\Lambda^\perp(\boldsymbol{F}_{\boldsymbol{B}_i})$ such that $||\tilde{\boldsymbol{T}}_{\boldsymbol{F}_{B_i}}|| = ||\tilde{\boldsymbol{T}}_{\boldsymbol{A}}||$. With input a bsis $\boldsymbol{T}_{\boldsymbol{F}_{\boldsymbol{B}_i}}$ of a lattice $\Lambda \subseteq \mathbb{R}^{m+m'}$, $s \geq L \cdot \omega\left(\sqrt{\log(m+m')}\right)$ and a center vector $\boldsymbol{0} \in \mathbb{R}^n$, the $\mathsf{SampleD}$ algorithm outputs a vector $\boldsymbol{d}_{\boldsymbol{B}_i} \in \Lambda^\perp(\boldsymbol{F}_{\boldsymbol{B}_i})$ from a distribution statistically close to $D_{\Lambda, s, \boldsymbol{0}}$. From Lemma 2.1, ℓ_2 norn of vectors sampled from the discrete Gaussian distribution has upper bound $s\sqrt{m+m'}$. This concludes that an honestly produced witness will always be accepted.

5.1.2 Security Analysis

Now we reduce the SIS problem to break the collision-resistance property of the accumulator scheme.

Theorem 5.1. *The accumulator construction of Figure 5.1 is collision-resistant under the hardness of* SIS *problem.*

Proof. Let $\mathcal{A}$ be a PPT adversary can break soundness property of the one-shot accumulator, we show a construction of a PPT algorithm $\mathcal{B}$ that can solve the SIS problem as follows:

1. $\mathcal{B}$ sets the public key pk of the accumulator scheme to $\boldsymbol{A}$.
2. $\mathcal{B}$ picks Q short random matrices $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_Q \in \mathbb{Z}^{m+m'}$ by independently sampling the columns of each matrix from $D_{\mathbb{Z}^m, s', \mathbf{0}}$ for some $s' \geq \omega\sqrt{\lg m}$ such that $||\boldsymbol{R}_i|| \leq s'\sqrt{m}$.
3. $\mathcal{B}$ sets the message space $\mathcal{M}$ where for $i \in [Q]$, $\boldsymbol{B}_i = \boldsymbol{A}\boldsymbol{R}_i \bmod q$. The distribution is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$.
4. $\mathcal{B}$ sends $(\boldsymbol{A}, \mathcal{M})$ to $\mathcal{A}$.
5. If $\mathcal{A}$ breaks the collision-resistance property of the accumulator, he can output a set $Y = \{\boldsymbol{B}_1^{(Y)}, \ldots, \boldsymbol{B}_k^{(Y)}\} \subset \mathcal{M}$, an element $\boldsymbol{B}^* \in \mathcal{M} \setminus Y$ and a witness $w^* \in \mathbb{Z}^{m+m'}$ such that
$$w^* \in \Lambda^{\perp}(\boldsymbol{A}||(v_Y - \boldsymbol{B}^*)), \text{ and } 0 < ||w^*|| \leq s\sqrt{m+m'}.$$
6. Compute $\boldsymbol{R}' = \sum_{j=1}^{k} \boldsymbol{R}_j^{(Y)} - \boldsymbol{R}^*$ where $\boldsymbol{R}_j^{(Y)}$ and $\boldsymbol{R}^*$ are the matrices picked by $\mathcal{B}$ to set $\boldsymbol{B}_j^{(Y)}$ and $\boldsymbol{B}^*$ respectively. Write $w^* \in \mathbb{Z}^{m+m'}$ as $\begin{bmatrix} w_1^* \\ w_2^* \end{bmatrix}$ such that $w_1^* \in \mathbb{Z}^m$ and $w_2^* \in \mathbb{Z}^{m'}$.
7. Eventually $\mathcal{B}$ outputs $\boldsymbol{e} = w_1^* + \boldsymbol{R}' w_2^* \in \mathbb{Z}^m$ as a solution to the SIS instance $(q, \boldsymbol{A} \in \mathbb{Z}_q^{n\times m}, \beta)$.

It is clear that

$$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{A}\left(w_1^* + \boldsymbol{R}' w_2^*\right) = \boldsymbol{A}w_1^* + \left(\sum_{j=1}^{k} \boldsymbol{A}\boldsymbol{R}_j^{(Y)} - \boldsymbol{A}\boldsymbol{R}^*\right) w_2^*$$

$$= \boldsymbol{A}w_1^* + \left(\sum_{j=1}^{k} \boldsymbol{B}_j^{(Y)} - \boldsymbol{B}^*\right) w_2^* = \left[\boldsymbol{A}||\left(v_Y - \boldsymbol{B}^*\right)\right] w^* = \mathbf{0} \bmod q.$$

Since $||\boldsymbol{R}_i|| \leq s'\sqrt{m}$ and $0 < ||w^*|| \leq s\sqrt{m+m'}$, we have

$$\begin{aligned}
\|\boldsymbol{e}\| = \left\|w_1^* + \boldsymbol{R}' w_2^*\right\| &= \left\|w_1^* + \left(\sum_{j=1}^{k} \boldsymbol{A}\boldsymbol{R}_j^{(Y)} - \boldsymbol{A}\boldsymbol{R}^*\right) w_2^*\right\| \\
&\leq \|w_1^*\| + \left\|\sum_{j=1}^{k} \boldsymbol{A}\boldsymbol{R}_j^{(Y)} - \boldsymbol{A}\boldsymbol{R}^*\right\| \|w_2^*\| \\
&\leq s\sqrt{m+m'}\left(1 + (k+1)s'\sqrt{m}\right) \\
&\leq Qs's\left(m+m'\right).
\end{aligned}$$

Hence, $\boldsymbol{e}$ is a valid solution to the SIS instance. □

5.2 Tree-style Accumulator from Lattices

The first trapdoor-less accumulator from lattices is claimed based on a Merkle hash tree [LLNW16], where the accumulated value is represented by the root, while the witness is represented by a path from the secret leaf to the root.

5.2.1 Accumulator Construction

We start by describing the lattice-based collision-resistant hash functions, on which the Merkle hash tree will be built.

We define the "powers-of-2" matrix

$$\boldsymbol{G} = \begin{bmatrix} 1\ 2\ 4\ \ldots\ 2^{k-1} & & & \\ & 1\ 2\ 4\ \ldots\ 2^{k-1} & & \\ & & \ddots & \\ & & & 1\ 2\ 4\ \ldots\ 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}.$$

where for every $\boldsymbol{v} \in \mathbb{Z}_q^n$, it holds that $\boldsymbol{v} = \boldsymbol{G} \cdot \mathsf{bin}(\boldsymbol{v})$ where $\mathsf{bin}(\boldsymbol{v}) \in \{0,1\}^{nk}$ is the binary representation of $\boldsymbol{v}$. In the rest of this section, we denote $m = 2nk$.

Definition 5.1. The hash function $h_{\boldsymbol{A}} : \{0,1\}^{nk} \times \{0,1\}^{nk} \mapsto \{0,1\}^{nk}$, described by a matrix $\boldsymbol{A} = [\boldsymbol{A}_0 \mid \boldsymbol{A}_1]$ with $\boldsymbol{A}_0, \boldsymbol{A}_1 \in \mathbb{Z}_q^{n \times nk}$, is defined as

$$h_{\boldsymbol{A}}(\boldsymbol{u}_0, \boldsymbol{u}_1) = \mathsf{bin}(\boldsymbol{A}_0 \cdot \boldsymbol{u}_0 + \boldsymbol{A}_1 \cdot \boldsymbol{u}_1 \bmod q) \in \{0,1\}^{nk}.$$

Not that

$$h_{\boldsymbol{A}}(\boldsymbol{u}_0, \boldsymbol{u}_1) = \boldsymbol{u} \Leftrightarrow \boldsymbol{A}_0 \cdot \boldsymbol{u}_0 + \boldsymbol{A}_1 \cdot \boldsymbol{u}_1 = \boldsymbol{G} \cdot \boldsymbol{u} \bmod q \tag{5.1}$$

Then, we show that the collision-resistance property follows from the hardness of worst-case approximation problem.

Lemma 5.1. *The hash function $h_{\boldsymbol{A}}$ in Definition 5.1 is collision-resistant under the hardness of the SIVP_γ problem.*

Proof. We are given a collision-finder C, which on input $\boldsymbol{A} = [\boldsymbol{A}_0 \mid \boldsymbol{A}_1] \leftarrow\!\$\, \mathbb{Z}_q^{n \times m}$ outputs two distinct pairs $(\boldsymbol{u}_0, \boldsymbol{u}_1) \in \{0,1\}^{nk} \times \{0,1\}^{nk}$ and $(\boldsymbol{v}_0, \boldsymbol{v}_1) \in \{0,1\}^{nk} \times \{0,1\}^{nk}$ such that $h_{\boldsymbol{A}}(\boldsymbol{u}_0, \boldsymbol{u}_1) = h_{\boldsymbol{A}}(\boldsymbol{v}_0, \boldsymbol{v}_1)$.

With the pair, we can get a non-zero vector $\boldsymbol{z} = \begin{pmatrix} \boldsymbol{u}_0 - \boldsymbol{v}_0 \\ \boldsymbol{u}_1 - \boldsymbol{v}_1 \end{pmatrix} \in \{-1, 0, 1\}^m$ such that

$$\boldsymbol{A} \cdot \boldsymbol{z} = \boldsymbol{A}_0 \cdot (\boldsymbol{u}_0 - \boldsymbol{v}_0) + \boldsymbol{A}_0 \cdot (\boldsymbol{u}_1 - \boldsymbol{v}_1) = \boldsymbol{0} \bmod q$$

which implies $\boldsymbol{z}$ is a valid solution to the $\mathsf{SIS}_{n,m,q,1}$ problem with the matrix $\boldsymbol{A}$. The lemma then follows from the worst-case to average-case reduction from SIVP_γ. □

The construction of the Merkle-tree style accumulator from lattices is shown in Figure 5.2. The tree has $N = 2^\ell$ leaves, where ℓ is a positive integer. At a high level, the accumulated value is the root node of the Merkle-tree. The witness is the index of the element in the set, along with the adjacent nodes of

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Return $\mathsf{pp} = \boldsymbol{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{n\times m}$
- $\mathsf{Acc}(\mathsf{pp}, S) \to v_S$: Without loss of generality, we denote the elements of the set S as $S = \{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{N-1}\}$. For all $j \in [0, N-1]$, i.e., the index of the elements, let $[j_1 \ldots j_\ell]$ be the binary representation of j, and let $\boldsymbol{d}_j = \boldsymbol{u}_{[j_1 \ldots j_\ell]}$ be the leaf nodes. We construct the Merkle tree as follows:

  1. At depth $i \in [\ell - 1]$, the nodes $\boldsymbol{u}_{b_1 \ldots b_i}$ for all $[b_1 \ldots b_i] \in \{0,1\}^i$ is defined as $h_{\boldsymbol{A}}(\boldsymbol{u}_{[b_1 \ldots b_i 0]}, \boldsymbol{u}_{[b_1 \ldots b_i 1]})$.
  2. At depth 0, the root $\boldsymbol{u} \in \{0,1\}^{nk}$ is defined as $h_{\boldsymbol{A}}(\boldsymbol{u}_{[0]}, \boldsymbol{u}_{[1]})$.

  Return the accumulated value as root value $\boldsymbol{u}$.
- $\mathsf{Wit}(\mathsf{pp}, S, \boldsymbol{d}_i) \to w$: Return the witness for $\boldsymbol{d}_i \in S$ as

$$w = \left([j_1 \ldots j_\ell], \left(\boldsymbol{u}_{[j_1 \ldots j_{\ell-1} \bar{j_\ell}]}, \ldots, \boldsymbol{u}_{[j_1 \bar{j_2}]}, \boldsymbol{u}_{[\bar{j_1}]}\right)\right) \in \{0,1\}^\ell \times \left(\{0,1\}^{nk}\right)^\ell.$$

- $\mathsf{Ver}(\mathsf{pp}, v_S, \boldsymbol{d}, w) \to \texttt{accept}/\texttt{reject}$: Let the witness be of the form

$$w = ([j_1 \ldots j_\ell], (\boldsymbol{w}_\ell, \ldots, \boldsymbol{w}_1)) \in \{0,1\}^\ell \times \left(\{0,1\}^{nk}\right)^\ell.$$

  The path $\boldsymbol{v}_\ell, \ldots, \boldsymbol{v}_1, \boldsymbol{v}_0$ can be computed as follows: $\boldsymbol{v}_\ell = \boldsymbol{d}$ and

$$\forall i \in \{\ell - 1, \ldots, 1, 0\} : \boldsymbol{v}_i = \begin{cases} h_{\boldsymbol{A}}(\boldsymbol{v}_{i+1}, \boldsymbol{w}_{i+1}), & \text{if } j_{i+1} = 0 \\ h_{\boldsymbol{A}}(\boldsymbol{w}_{i+1}, \boldsymbol{v}_{i+1}), & \text{if } j_{i+1} = 1 \end{cases}$$

  If $\boldsymbol{v}_0 = \boldsymbol{u}$, then return `accept`, otherwise, return `reject`.

Figure 5.2: Merkle-tree style Accumulator from Lattices

the ones constructing a path. To verify whether a witness is valid for an element, the algorithm gets the index of the element and then construct the path from leaf to root with the adjacent nodes provided in the witness. The witness is valid if the path arrives at the correct root node.

We illustrate the construction by an example of a tree with $2^3 = 8$ leaves, shown in Figure 5.3. In this accumulator, a set of 8 elements $S = \{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_7\}$ is accumulated, represented by the root node $\boldsymbol{u}$. The witness of the element $\boldsymbol{d}_5$ is the element index [101] along with the gray nodes $(\boldsymbol{u}_{[100]}, \boldsymbol{u}_{[11]}, \boldsymbol{u}_{[0]})$.

To verify a witness $\left([101], (\boldsymbol{u}_{[100]}, \boldsymbol{u}_{[11]}, \boldsymbol{u}_{[0]})\right)$ is a valid one, we compute the path as follow: let $\boldsymbol{v}_3 = \boldsymbol{d}_5$ (i.e., $\boldsymbol{u}_{[101]}$).

Given $j_3 = 1$, $\boldsymbol{v}_2 = h_{\boldsymbol{A}}\left(\boldsymbol{w}_{[3]}, \boldsymbol{v}_3\right) = h_{\boldsymbol{A}}\left(\boldsymbol{u}_{[100]}, \boldsymbol{u}_{[101]}\right)$, which is $\boldsymbol{u}_{[10]}$ indeed. Then, given $j_2 = 0$, $\boldsymbol{v}_1 = h_{\boldsymbol{A}}\left(\boldsymbol{v}_{[2]}, \boldsymbol{w}_2\right) = h_{\boldsymbol{A}}\left(\boldsymbol{u}_{[10]}, \boldsymbol{u}_{[11]}\right)$, which equals $\boldsymbol{u}_{[1]}$. Finally, with $j_1 = 1$, $\boldsymbol{v}_0 = h_{\boldsymbol{A}}\left(\boldsymbol{w}_{[1]}, \boldsymbol{v}_1\right) = h_{\boldsymbol{A}}\left(\boldsymbol{u}_{[0]}, \boldsymbol{u}_{[1]}\right)$.

Since $\boldsymbol{v}_0 = \boldsymbol{u}$, the witness is indeed a valid one for $\boldsymbol{d}_5$. The red nodes show

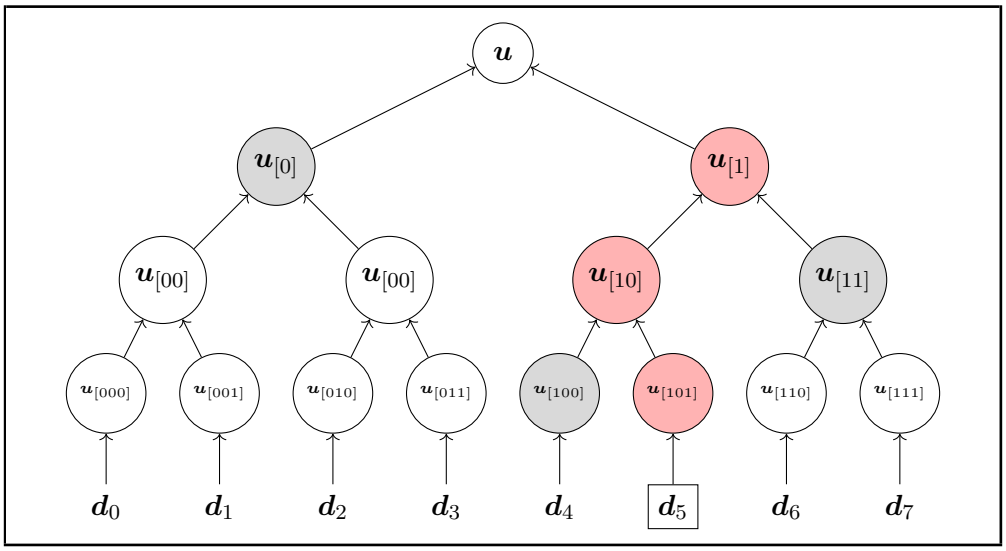

Figure 5.3: A Merkle-tree style accumulator from lattices with $2^3 = 8$ leaves.

the path computed in the verification process.

5.2.2 Security Analysis

It is straightforward to see that the accumulator scheme is correct. Then we show that the security is based on the collision-resistance property of the hash function.

Theorem 5.2. *The accumulator construction of Figure 5.2 is collision-resistant under the hardness of the* SIVP_γ *problem.*

Proof. Let $\mathcal{A}$ be a PPT adversary that can break the collision-resistance property of the accumulator, which means that he can output a set $Y = (\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{N-1})$, a value $\boldsymbol{d}^* \notin Y$ and a witness w^* such that $(\mathsf{pp}, v_Y, \boldsymbol{d}^*, w^*)$ can be accepted by the verification algorithm. We show that this can be used to break the collision-resistance property of the hash function in Definition 5.1.

Parse $w^* = ([j_1^* \ldots j_\ell^*], (\boldsymbol{w}_\ell^*, \ldots, \boldsymbol{w}_1^*))$ where $[j_1^* \ldots j_\ell^*]$ is the binary representation of j^* (note that since the witness is accepted, j^* should be the index of some elements involved in the tree construction and $\boldsymbol{d}_{j^*}$ should be some leaf node). With the Acc algorithm, we compute the path from the leaf node to the root, i.e., $\boldsymbol{u}_{[j_1^* \ldots j_\ell^*]}, \ldots, \boldsymbol{u}_{j_1^*}, \boldsymbol{u}^*$. On the other hand, the Ver algorithm computes the path, i.e., $\boldsymbol{v}_\ell^*, \ldots, \boldsymbol{v}_1^*, \boldsymbol{v}_0^*$.

Since the witness is accepted, we have $\boldsymbol{u}^* = \boldsymbol{v}_0^*$. However, $\boldsymbol{d}^* \neq \boldsymbol{d}_{j^*}$ since $\boldsymbol{d} \notin Y$. Then, comparing the two paths, once we find the smallest integer $k \in [\ell]$ such that $\boldsymbol{v}_k^* \neq \boldsymbol{u}_{[j_1^* \ldots j_k^*]}$, we can get a collision for $h_{\boldsymbol{A}}$ at the parent node of $\boldsymbol{u}_{[j_1^* \ldots j_k^*]}$.

The theorem then follows from Lemma 5.1 that $h_{\boldsymbol{A}}$ is collision-resistant under the hardness of the SIVP_γ problem. □

5.2.3 Zero-knowledge Proof of Accumulated Values

A zero-knowledge interactive proof system for the accumulated value can be constructed in Stern's style [Ste94a] with extension technique [LNSW13].

In the protocol, on input $(\boldsymbol{A}, \boldsymbol{u})$, the prover P proves to the verifier V that P owns a value-witness pair $(\boldsymbol{d}, w)$ which is accepted by the verification algorithm Ver to be valid. The relation R is defined as

$$R = \left\{ \left((\boldsymbol{A}, \boldsymbol{u}) \in \mathbb{Z}_q^{n \times m} \times \{0,1\}^{nk}; \boldsymbol{d} \in \{0,1\}^{nk}, w \in \{0,1\}^{\ell} \times \left(\{0,1\}^{nk}\right)^{\ell} \right) : \mathsf{Ver}(\boldsymbol{A}, \boldsymbol{u}, \boldsymbol{d}, w) = 1 \right\}$$

Let $(\boldsymbol{d}, w)$ be a valid value-witness pair, where $w = ([j_1 \dots j_\ell], (\boldsymbol{w}_\ell, \dots, \boldsymbol{w}_1))$. Let $\boldsymbol{v}_\ell, \dots, \boldsymbol{v}_1, \boldsymbol{v}_0$ be the path computed by the verification algorithm Ver. Recall that if the value-witness pair is accepted, we have $\boldsymbol{v}_\ell = \boldsymbol{d}$, $\boldsymbol{v}_0 = \boldsymbol{u}$ and

$$\forall i \in \{\ell - 1, \dots, 1, 0\} : \boldsymbol{v}_i = \begin{cases} h_{\boldsymbol{A}}(\boldsymbol{v}_{i+1}, \boldsymbol{w}_{i+1}), & \text{if } j_{i+1} = 0 \\ h_{\boldsymbol{A}}(\boldsymbol{w}_{i+1}, \boldsymbol{v}_{i+1}), & \text{if } j_{i+1} = 1 \end{cases} \tag{5.2}$$

Since j_{i+1} can only be 1 or 0, we can write Equation 5.2 in a compact form:

$$\forall i \in \{\ell - 1, \dots, 1, 0\} : \boldsymbol{v}_i = \bar{j}_{i+1} \cdot h_{\boldsymbol{A}}(\boldsymbol{v}_{i+1}, \boldsymbol{w}_{i+1}) + j_{i+1} \cdot h_{\boldsymbol{A}}(\boldsymbol{w}_{i+1}, \boldsymbol{v}_{i+1}) \tag{5.3}$$

By Equation 5.1, Equation 5.3 is equivalent to $\forall i \in \{\ell - 1, \dots, 1, 0\}$:

$$\begin{aligned} & \bar{j}_{i+1} \cdot (\boldsymbol{A}_0 \cdot \boldsymbol{v}_{i+1} + \boldsymbol{A}_1 \cdot \boldsymbol{w}_{i+1}) + j_{i+1} \cdot (\boldsymbol{A}_0 \cdot \boldsymbol{w}_{i+1} + \boldsymbol{A}_1 \cdot \boldsymbol{v}_{i+1}) = \boldsymbol{G} \cdot \boldsymbol{v}_i \bmod q \\ \Leftrightarrow & \boldsymbol{A} \cdot \begin{pmatrix} \bar{j}_{i+1} \cdot \boldsymbol{v}_{i+1} \\ j_{i+1} \cdot \boldsymbol{v}_{i+1} \end{pmatrix} + \boldsymbol{A} \cdot \begin{pmatrix} j_{i+1} \cdot \boldsymbol{w}_{i+1} \\ \bar{j}_{i+1} \cdot \boldsymbol{w}_{i+1} \end{pmatrix} = \boldsymbol{G} \cdot \boldsymbol{v}_i \bmod q \end{aligned}$$

Let $\mathsf{ext}(b, \boldsymbol{v})$ denote the vector $\boldsymbol{z} = \begin{pmatrix} \bar{b} \cdot \boldsymbol{v} \\ b \cdot \boldsymbol{v} \end{pmatrix}$ for $i \in \{nk, m\}$, $b \in \{0,1\}$ and $\boldsymbol{v} \in \{0,1\}^i$. Moreover, the equation is equivalent to

$$\boldsymbol{A} \cdot \mathsf{ext}\left(j_{i+1}, \boldsymbol{v}_{i+1}\right) + \boldsymbol{A} \cdot \mathsf{ext}\left(\bar{j_{i+1}}, \boldsymbol{w}_{i+1}\right) = \boldsymbol{G} \cdot \boldsymbol{v}_i \bmod q. \tag{5.4}$$

Therefore, the prover P needs to convince the verifier V that he knows $[j_1 \dots j_\ell] \in \{0,1\}^\ell$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_\ell, \boldsymbol{w}_1, \dots, \boldsymbol{w}_\ell \in \{0,1\}^{nk}$ such that

$$\begin{cases} \boldsymbol{A} \cdot \mathsf{ext}\left(j_1, \boldsymbol{v}_1\right) + \boldsymbol{A} \cdot \mathsf{ext}\left(\bar{j_1}, \boldsymbol{w}_1\right) = \boldsymbol{G} \cdot \boldsymbol{u} \bmod q \\ \forall i \in [\ell - 1] : \boldsymbol{A} \cdot \mathsf{ext}\left(j_{i+1}, \boldsymbol{v}_{i+1}\right) + \boldsymbol{A} \cdot \mathsf{ext}\left(\bar{j_{i+1}}, \boldsymbol{w}_{i+1}\right) = \boldsymbol{G} \cdot \boldsymbol{v}_i \bmod q. \end{cases} \tag{5.5}$$

Note that since $\boldsymbol{v_0} = \boldsymbol{u}$ which is the public input, it is separated from the cases for $i \in [\ell - 1]$.

To apply the Stern's protocol shown in Figure 3.2, some extension technique has been applied here. In Stern's protocol, the secret $\boldsymbol{x}$, corresponding to the 2ℓ vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_\ell, \boldsymbol{w}_1, \dots, \boldsymbol{w}_\ell \in \{0,1\}^{nk}$ here, should be sampled from $\mathsf{B}(m, m/2)$. Note that here we have $m = 2nk$, and thus we extend the 2ℓ vectors into $\boldsymbol{v}_1^*, \dots, \boldsymbol{v}_\ell^*, \boldsymbol{w}_1^*, \dots, \boldsymbol{w}_\ell^* \in \mathsf{B}(m, nk)$ respectively. This is done by appending a length-nk vector of suitable Hamming weight to each of the vectors.

For the public input $\boldsymbol{A}$ and "power-of-2" matrix $\boldsymbol{G}$, we extend $\boldsymbol{A} = [\boldsymbol{A}_0 \mid \boldsymbol{A}_1]$ to matrix $\boldsymbol{A}^* = \left[\boldsymbol{A}_0 \mid \boldsymbol{0}^{n \times nk} \mid \boldsymbol{A}_1 \mid \boldsymbol{0}^{n \times nk}\right] \in \mathbb{Z}_q^{n \times 2m}$, and extend $\boldsymbol{G}$ into matrix $\boldsymbol{G}^* = \left[\boldsymbol{G} \mid \boldsymbol{0}^{n \times nk}\right] \in \mathbb{Z}_q^{n \times m}$.

Now, let $\boldsymbol{z}_i = \mathsf{ext}(j_i, \boldsymbol{v}_i^*)$ and $\boldsymbol{y}_i = \mathsf{ext}(\bar{j}_i, \boldsymbol{w}_i^*)$ for all $i \in [\ell]$. What the prover P wants to convince, Equation 5.5, can be rewritten as

$$\begin{cases} \boldsymbol{A}^* \cdot \boldsymbol{z}_1 + \boldsymbol{A}^* \cdot \boldsymbol{y}_1 = \boldsymbol{G} \cdot \boldsymbol{u} \bmod q \\ \forall i \in [\ell - 1] : \boldsymbol{A}^* \cdot \boldsymbol{z}_{i+1} + \boldsymbol{A}^* \cdot \boldsymbol{y}_{i+1} = \boldsymbol{G}^* \cdot \boldsymbol{v}_i^* \bmod q \end{cases} \tag{5.6}$$

Another essential component in Stern's protocol is the permutation π. It is straight forward that for each vector we need one independent permutation. However, this is not enough due to the use of $\mathsf{ext}(b, \boldsymbol{v})$.

For $b \in \{0,1\}$ and $\pi \in \mathcal{S}_m$, let $F_{b,\pi}$ be the permutation that transforms $\boldsymbol{z} = \begin{pmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \end{pmatrix} \in \mathbb{Z}_q^{2m}$ into $F_{b,\pi} = \begin{pmatrix} \pi(\boldsymbol{z}_b) \\ \pi(\boldsymbol{z}_{\bar{b}}) \end{pmatrix}$.

The permutations has the following equivalences: For all $c, b \in \{0,1\}$, all permutations $\pi, \phi \in \mathcal{S}_m$ and all $\boldsymbol{v}, \boldsymbol{w} \in \{0,1\}^m$,

$$\begin{cases} \boldsymbol{z} = \mathsf{ext}(c, \boldsymbol{v}) \wedge \boldsymbol{v} \in \mathsf{B}(m, nk) \Leftrightarrow F_{b,\pi}(\boldsymbol{z}) = \mathsf{ext}(c \oplus b, \pi(\boldsymbol{v})) \wedge \pi(\boldsymbol{v}) \in \mathsf{B}(m, nk) \\ \boldsymbol{y} = \mathsf{ext}(\bar{c}, \boldsymbol{w}) \wedge \boldsymbol{w} \in \mathsf{B}(m, nk) \Leftrightarrow F_{\bar{b},\phi}(\boldsymbol{y}) = \mathsf{ext}(c \oplus b, \phi(\boldsymbol{w})) \wedge \phi(\boldsymbol{w}) \in \mathsf{B}(m, nk) \end{cases} \tag{5.7}$$

With the extension technique and permutations, we can apply Stern's protocol here. The interactive proof system is shown in Figure 5.4.

Theorem 5.3. *The protocol in Figure 5.4 is statistically zero knowledge when the commitment scheme* Com *is statistically-hiding and computationally-binding.*

Proof. This proof is similar to the one of the Stern's protocol (Theorem 3.1). The permutations and 2ℓ elements of the private input make this proof much more complex. Different from the proof style for Theorem 3.1, we omit the same components among the cases.

We construct a simulator Sim on input $\boldsymbol{A}$ and $\boldsymbol{u}$ and given oracle access to a (cheating) verifier V^* outputting a simulated transcript, which is statistically indistinguishable from the distribution $\mathsf{view}(\langle \mathsf{P}(\boldsymbol{A}, \boldsymbol{u}), \mathsf{V}^*(\boldsymbol{A}) \rangle)$.

The simulator Sim chooses a random value $\mathsf{ch}^* \in \{1,2,3\}$ for the challenge that the cheating verifier V^* will *not* choose. The random tape is implicit.

Case $\mathsf{ch}^* = 1$**:** Using linear algebra, the simulator Sim computes $\boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_{\ell-1} \in \mathbb{Z}_q^m$, $\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_\ell, \boldsymbol{y}'_1, \ldots, \boldsymbol{y}'_\ell \in \mathbb{Z}_q^{2m}$ such that

$$\begin{cases} \boldsymbol{A}^* \cdot \boldsymbol{z}'_1 + \boldsymbol{A}^* \cdot \boldsymbol{y}'_1 = \boldsymbol{G} \cdot \boldsymbol{u} \bmod q \\ \forall i \in [1, \ell - 1] : \boldsymbol{A}^* \cdot \boldsymbol{z}'_{i+1} + \boldsymbol{A}^* \cdot \boldsymbol{y}'_{i+1} = \boldsymbol{G}^* \cdot \boldsymbol{v}'_i \bmod q \end{cases}.$$

It samples random permutations and random strings

$$\begin{cases} b_1, \ldots, b_\ell \leftarrow\$ \{0,1\}; \pi_1, \ldots, \pi_\ell, \phi_1, \ldots, \phi_\ell \leftarrow\$ \mathcal{S}_m \\ \boldsymbol{r}_{\boldsymbol{v}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{v}}^{(\ell-1)} \leftarrow\$ \mathbb{Z}_q^m; \boldsymbol{r}_{\boldsymbol{z}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{z}}^{(\ell)}, \boldsymbol{r}_{\boldsymbol{y}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{y}}^{(\ell)} \leftarrow\$ \mathbb{Z}_q^{2m} \end{cases}$$

and random strings for the commitment scheme $\rho'_1, \rho'_2, \rho'_3$. Then, it sends the commitments to V^* computed as follows

$$\begin{aligned} \mathsf{com}'_1 = \mathsf{Com}(\{b_i; \pi_i; \phi_i\}_{i=1}^{\ell}; \boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{z}}^{(1)} + \boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{y}}^{(1)}; \\ \{\boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{z}}^{(i+1)} + \boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{y}}^{(i+1)} - \boldsymbol{G}^* \cdot \boldsymbol{r}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \rho'_1) \end{aligned}$$

– $\mathsf{com}'_2 = \mathsf{Com}(\{\pi_i(\boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{F_{b_i,\pi_i}(\boldsymbol{r}_{\boldsymbol{z}}^{(i)}); F_{\overline{b}_i,\phi_i}(\boldsymbol{r}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell}; \rho'_2)$,

– $\mathsf{com}'_3 = \mathsf{Com}(\{\pi_i(\boldsymbol{v}'_i + \boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1};$
$\{F_{b_i,\pi_i}(\boldsymbol{z}'_i + \boldsymbol{r}_{\boldsymbol{z}}^{(i)}); F_{\overline{b}_i,\phi_i}(\boldsymbol{y}'_i + \boldsymbol{r}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell}; \rho'_3)$.

Upon a challenge ch from V^*, the simulator Sim outputs a transcript as follows:

– If $\mathsf{ch} = 1$, Sim outputs $\perp$ and halts.

– If $\mathsf{ch} = 2$, Sim outputs

$$((\mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3), 2, (\{\pi_i(\boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{F_{b_i,\pi_i}(\boldsymbol{r}_{\boldsymbol{z}}^{(i)}); F_{\overline{b}_i,\phi_i}(\boldsymbol{r}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell}; \rho_2))$$

.

– If $\mathsf{ch} = 3$, Sim outputs

$$((\mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3), 3, (\{\pi_i(\boldsymbol{v}'_i + \boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{F_{b_i,\pi_i}(\boldsymbol{z}'_i + \boldsymbol{r}_{\boldsymbol{z}}^{(i)}); F_{\overline{b}_i,\phi_i}(\boldsymbol{y}'_i + \boldsymbol{r}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell}; \rho_3))$$

.

Case $\mathsf{ch}^* = 2$: The simulator samples random permutations and random vectors

$$\begin{cases} j'_1, \ldots, j'_\ell \leftarrow\$ \{0,1\}; \boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_\ell, \boldsymbol{w}'_1, \ldots, \boldsymbol{w}'_\ell \leftarrow\$ \mathsf{B}(m, nk) \\ b_1, \ldots, b_\ell \leftarrow\$ \{0,1\}; \pi_1, \ldots, \pi_\ell, \phi_1, \ldots, \phi_\ell \leftarrow\$ \mathcal{S}_m \\ \boldsymbol{r}_{\boldsymbol{v}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{v}}^{(\ell-1)} \leftarrow\$ \mathbb{Z}_q^m; \boldsymbol{r}_{\boldsymbol{z}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{z}}^{(\ell)}, \boldsymbol{r}_{\boldsymbol{y}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{y}}^{(\ell)} \leftarrow\$ \mathbb{Z}_q^{2m} \end{cases}$$

and random strings for the commitment scheme $\rho'_1, \rho'_2, \rho'_3$. It computes $\boldsymbol{z}'_i = \mathsf{ext}(j'_i, \boldsymbol{v}'_i), \boldsymbol{y}'_i = \mathsf{ext}(\overline{j'_i}, \boldsymbol{w}'_i)$ for each $i \in [\ell]$. Then, it sends the commitments to V^* computed in the same way as case $\mathsf{ch}^* = 1$.

Upon a challenge ch from V^*, the simulator Sim outputs a transcript as follows:

– If $\mathsf{ch} = 1$, Sim outputs

$$((\mathsf{com}'_1, \mathsf{com}'_2, \mathsf{com}'_3), 1, (\{\pi_i(\boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{j'_i \oplus b_i; \pi_i(\boldsymbol{v}'_i); F_{b_i,\pi_i}(\boldsymbol{r}_{\boldsymbol{z}}^{(i)}); \phi_i(\boldsymbol{w}'_i); F_{\overline{b}_i,\phi_i}(\boldsymbol{r}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell}; \rho'_2; \rho'_3))$$

.

– If $\mathsf{ch} = 2$, $\perp$ and halts.

– If $\mathsf{ch} = 3$, Sim outputs the same transcript as the case $\mathsf{ch} = 3$ under $\mathsf{ch}^* = 1$.

Case $\mathsf{ch}^* = 3$: The simulator samples and computes the same things as case $\mathsf{ch}^* = 2$ except that

$$\mathsf{com}'_1 = \mathsf{Com}(\{b_i; \pi_i; \phi_i\}_{i=1}^{\ell}; \boldsymbol{A}^* \cdot (\boldsymbol{z}'_1 + \boldsymbol{r}_{\boldsymbol{z}}^{(1)}) + \boldsymbol{A}^* \cdot (\boldsymbol{y}'_1 + \boldsymbol{r}_{\boldsymbol{y}}^{(1)}) - \boldsymbol{G} \cdot \boldsymbol{u}; \{\boldsymbol{A}^* \cdot (\boldsymbol{z}'_{i+1} + \boldsymbol{r}_{\boldsymbol{z}}^{(i+1)}) + \boldsymbol{A}^* \cdot (\boldsymbol{y}'_{i+1} + \boldsymbol{r}_{\boldsymbol{y}}^{(i+1)}) - \boldsymbol{G}^* \cdot (\boldsymbol{v}'_i + \boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \rho'_1)$$

Since the commitment scheme used is statistically hiding, the distribution of the commitments and the distribution of the challenge from the cheating verifier V^* in the simulated view are statistically indistinguishable from the ones in the real protocol. By the above arguments, the distribution of the simulator's output conditioned on it is not $\perp$, is statistically indistinguishable from the view in the real protocol. □

Then, we show that the protocol in Figure 5.4 has 3-special soundness property.

Lemma 5.2. *If the commitment scheme* Com *is statistically hiding and computationally binding, then there exists an efficient knowledge extract* Ext *which on input* 3 *valid responses* $(\mathsf{res}_1, \mathsf{res}_2, \mathsf{res}_3)$ *to the same commitment series* $(\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$ *outputs a pair* $(\boldsymbol{d}', w')$ *such that* $((\boldsymbol{A}, \boldsymbol{u}); \boldsymbol{d}', w') \in R$.

Proof. Given the 3 valid responses to $(\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$,

$$\begin{cases} \mathsf{res}_1 = (\{\boldsymbol{t}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{a_i; \boldsymbol{s}_{\boldsymbol{v}}^{(i)}; \boldsymbol{t}_{\boldsymbol{z}}^{(i)}; \boldsymbol{s}_{\boldsymbol{w}}^{(i)}; \boldsymbol{t}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell}) \\ \mathsf{res}_2 = (\{\boldsymbol{e}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{c_i; \widehat{\pi}_i; \widehat{\phi}_i; \boldsymbol{e}_{\boldsymbol{z}}^{(i)}; \boldsymbol{e}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell}) \\ \mathsf{res}_3 = (\{\boldsymbol{p}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{d_i; \widetilde{\pi}_i; \widetilde{\phi}_i; \boldsymbol{p}_{\boldsymbol{z}}^{(i)}; \boldsymbol{p}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell}) \end{cases}$$

The validity of res_1 implies that

$$\forall i \in [\ell] : \boldsymbol{s}_{\boldsymbol{v}}^{(i)}, \boldsymbol{s}_{\boldsymbol{w}}^{(i)} \in \mathsf{B}(m, nk).$$

Since the 3 responses are all valid, from the binding property of the commitment scheme, for all $i \in [1, \ell-1]$: $\boldsymbol{t}_{\boldsymbol{v}}^{(i)} = \widetilde{\pi}_i(\boldsymbol{p}_{\boldsymbol{v}}^{(i)}); \boldsymbol{s}_{\boldsymbol{v}}^{(i)} + \boldsymbol{t}_{\boldsymbol{v}}^{(i)} = \widehat{\pi}_i(\boldsymbol{e}_{\boldsymbol{v}}^{(i)})$, and

$$\begin{cases} \boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{z}}^{(1)} + \boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{y}}^{(1)} - \boldsymbol{G} \cdot \boldsymbol{u} = \boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{z}}^{(1)} + \boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{y}}^{(1)} \bmod q \\ \forall i \in [1, \ell-1] : \boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{z}}^{(i+1)} + \boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{y}}^{(i+1)} - \boldsymbol{G}^* \cdot \boldsymbol{e}_{\boldsymbol{v}}^{(i)} \\ \qquad\qquad = \boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{z}}^{(i+1)} + \boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{y}}^{(i+1)} - \boldsymbol{G}^* \cdot \boldsymbol{p}_{\boldsymbol{v}}^{(i)} \bmod q \end{cases} \tag{5.8}$$

and for all $i \in [\ell]$:

$$\begin{cases} c_i = d_i; \widehat{\pi}_i = \widetilde{\pi}_i; \widehat{\phi}_i = \widetilde{\phi}_i; \\ \boldsymbol{t}_{\boldsymbol{z}}^{(i)} = F_{d_i, \widetilde{\pi}_i}(\boldsymbol{p}_{\boldsymbol{z}}^{(i)}); \mathsf{ext}(a_i, \boldsymbol{s}_{\boldsymbol{v}}^{(i)}) + \boldsymbol{t}_{\boldsymbol{z}}^{(i)} = F_{c_i, \widehat{\pi}_i}(\boldsymbol{e}_{\boldsymbol{z}}^{(i)}); \\ \boldsymbol{t}_{\boldsymbol{y}}^{(i)} = F_{\bar{d}_i, \widetilde{\phi}_i}(\boldsymbol{p}_{\boldsymbol{y}}^{(i)}); \mathsf{ext}(a_i, \boldsymbol{s}_{\boldsymbol{w}}^{(i)}) + \boldsymbol{t}_{\boldsymbol{y}}^{(i)} = F_{\bar{c}_i, \widehat{\phi}_i}(\boldsymbol{e}_{\boldsymbol{y}}^{(i)}). \end{cases}$$

The extractor Ext proceeds as follows. First, for $i \in [\ell]$, let

$$j_i = a_i \oplus c_i; \boldsymbol{v}_i^* = \widehat{\pi}_i^{-1}(\boldsymbol{s}_{\boldsymbol{v}}^{(i)}); \boldsymbol{w}_i^* = \widehat{\phi}_i^{-1}(\boldsymbol{s}_{\boldsymbol{w}}^{(i)}); \boldsymbol{z}_i = \boldsymbol{e}_{\boldsymbol{z}}^{(i)} - \boldsymbol{p}_{\boldsymbol{z}}^{(i)}; \boldsymbol{y}_i = \boldsymbol{e}_{\boldsymbol{y}}^{(i)} - \boldsymbol{p}_{\boldsymbol{y}}^{(i)}$$

.

Then, Equation 5.8 is equivalent to

$$\begin{cases} \boldsymbol{A}^* \cdot \boldsymbol{z}_1 + \boldsymbol{A}^* \cdot \boldsymbol{y}_1 = \boldsymbol{G} \cdot \boldsymbol{u} \bmod q \\ \forall i \in [1, \ell-1] : \boldsymbol{A}^* \cdot \boldsymbol{z}_{i+1} + \boldsymbol{A}^* \cdot \boldsymbol{y}_{i+1} = \boldsymbol{G}^* \cdot \boldsymbol{v}_i^* \bmod q \end{cases} \tag{5.9}$$

Note that

$$\begin{cases} \widehat{\pi}_i(\boldsymbol{v}_i^*) = \boldsymbol{s}_{\boldsymbol{v}}^{(i)} \in \mathsf{B}(m, nk) \wedge F_{c_i,\widehat{\pi}_i}(\boldsymbol{z}_i) = \mathsf{ext}(a_i, \boldsymbol{s}_{\boldsymbol{v}}^{(i)}) = \mathsf{ext}(j_i \oplus c_i, \widehat{\pi}_i(\boldsymbol{v}_i^*)) \\ \widehat{\phi}_i(\boldsymbol{w}_i^*) = \boldsymbol{s}_{\boldsymbol{w}}^{(i)} \in \mathsf{B}(m, nk) \wedge F_{\bar{c}_i,\widehat{\phi}_i}(\boldsymbol{y}_i) = \mathsf{ext}(a_i, \boldsymbol{s}_{\boldsymbol{w}}^{(i)}) = \mathsf{ext}(\bar{j}_i \oplus \bar{c}_i, \widehat{\phi}_i(\boldsymbol{w}_i^*)) \end{cases}$$

From Equation 5.7, it holds that

$$\begin{cases} \boldsymbol{v}_i^* \in \mathsf{B}(m, nk) \wedge \boldsymbol{z}_i = \mathsf{ext}(j_i, \boldsymbol{v}_i^*) \\ \boldsymbol{w}_i^* \in \mathsf{B}(m, nk) \wedge \boldsymbol{y}_i = \mathsf{ext}(\bar{j}_i, \boldsymbol{w}_i^*) \end{cases}$$

Then, Equation 5.9 is equivalent to

$$\begin{cases} \boldsymbol{A}^* \cdot \mathsf{ext}(j_1, \boldsymbol{v}_1^*) + \boldsymbol{A}^* \cdot \mathsf{ext}(\bar{j}_i, \boldsymbol{w}_i^*) = \boldsymbol{G} \cdot \boldsymbol{u} \bmod q \\ \forall i \in [1, \ell - 1] : \boldsymbol{A}^* \cdot \mathsf{ext}(j_{i+1}, \boldsymbol{v}_{i+1}^*) + \boldsymbol{A}^* \cdot \mathsf{ext}(\bar{j}_{i+1}, \boldsymbol{w}_{i+1}^*) = \boldsymbol{G}^* \cdot \boldsymbol{v}_i^* \bmod q \end{cases} \tag{5.10}$$

By dropping the last nk coordinates from $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_\ell^*, \boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_\ell^*$, the knowledge extractor Ext obtains $\boldsymbol{v}_1', \ldots, \boldsymbol{v}_\ell', \boldsymbol{w}_1', \ldots, \boldsymbol{w}_\ell' \in \{0,1\}^{nk}$ which satisfies

$$\begin{cases} \boldsymbol{A} \cdot \mathsf{ext}(j_1, \boldsymbol{v}_1') + \boldsymbol{A} \cdot \mathsf{ext}(\bar{j}_1, \boldsymbol{w}_1') = \boldsymbol{G} \cdot \boldsymbol{u} \bmod q \\ \forall i \in [1, \ell - 1] : \boldsymbol{A} \cdot \mathsf{ext}(j_{i+1}, \boldsymbol{v}_{i+1}') + \boldsymbol{A} \cdot \mathsf{ext}(\bar{j}_{i+1}, \boldsymbol{w}_{i+1}') = \boldsymbol{G} \cdot \boldsymbol{v}_i' \bmod q \end{cases} \tag{5.11}$$

which is what the prover P wants to convince the verifier V as in Equation 5.5.

Let $\boldsymbol{d}' = \boldsymbol{v}_\ell'$ and $w' = ([j_1, \ldots, j_\ell], (\boldsymbol{w}_\ell', \ldots, \boldsymbol{w}_1'))$, $(\boldsymbol{d}', w')$ is a valid pair that will be accepted by the verifier. □

Public parameters: n, q, k, m, ℓ, "powers-of-2" matrix $\boldsymbol{G}$ and extension $\boldsymbol{G}^*$.
P, V:
The common input is $(\boldsymbol{A}, \boldsymbol{u})$. Both extend $\boldsymbol{A}$ to $\boldsymbol{A}^*$. The prover P's auxiliary input is $([j_1, \ldots, j_\ell], (\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_\ell^*), (\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_\ell^*), (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_\ell), (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\ell))$. They interact as follows (the random tape and random strings for the commitment scheme are implicit):

- **Round 1** (com): The prover P choose the following permutations and random strings

$$\begin{cases} b_1, \ldots, b_\ell \leftarrow\$ \{0,1\}; \pi_1, \ldots, \pi_\ell, \phi_1, \ldots, \phi_\ell \leftarrow\$ \mathcal{S}_m \\ \boldsymbol{r}_{\boldsymbol{v}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{v}}^{(\ell-1)} \leftarrow\$ \mathbb{Z}_q^m; \boldsymbol{r}_{\boldsymbol{z}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{z}}^{(\ell)}, \boldsymbol{r}_{\boldsymbol{y}}^{(1)}, \ldots, \boldsymbol{r}_{\boldsymbol{y}}^{(\ell)} \leftarrow\$ \mathbb{Z}_q^{2m} \end{cases}$$

  and send commitments $\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3$ computed as

  - $\mathsf{com}_1 = \mathsf{Com}(\{b_i; \pi_i; \phi_i\}_{i=1}^{\ell}; \boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{z}}^{(1)} + \boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{y}}^{(1)}; \{\boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{z}}^{(i+1)} + \boldsymbol{A}^* \cdot \boldsymbol{r}_{\boldsymbol{y}}^{(i+1)} - \boldsymbol{G}^* \cdot \boldsymbol{r}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1})$
  - $\mathsf{com}_2 = \mathsf{Com}(\{\pi_i(\boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{F_{b_i,\pi_i}(\boldsymbol{r}_{\boldsymbol{z}}^{(i)}); F_{\overline{b}_i,\phi_i}(\boldsymbol{r}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell})$,
  - $\mathsf{com}_3 = \mathsf{Com}(\{\pi_i(\boldsymbol{v}_i^* + \boldsymbol{r}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{F_{b_i,\pi_i}(\boldsymbol{z}_i + \boldsymbol{r}_{\boldsymbol{z}}^{(i)}); F_{\overline{b}_i,\phi_i}(\boldsymbol{y}_i + \boldsymbol{r}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell})$.

- **Round 2** (ch): The verifier V sends a random challenge $\mathsf{ch} \in \{1, 2, 3\}$ to the prover P.

- **Round 3** (res): The prover P responses upon the challenge ch received:

  - If $\mathsf{ch} = 1$, reveal com_2 and com_3.
    For each $i \in [\ell - 1]$, let $\boldsymbol{t}_{\boldsymbol{v}}^{(i)} = \pi_i(\boldsymbol{r}_{\boldsymbol{v}}^{(i)})$;
    For each $i \in [\ell]$, let $a_i = j_i \oplus b_i; \boldsymbol{s}_{\boldsymbol{v}}^{(i)} = \pi_i(\boldsymbol{v}_i^*); \boldsymbol{s}_{\boldsymbol{w}}^{(i)} = \phi_i(\boldsymbol{w}_i^*); \boldsymbol{t}_{\boldsymbol{z}}^{(i)} = F_{b_i,\pi_i}(\boldsymbol{r}_{\boldsymbol{z}}^{(i)}); \boldsymbol{t}_{\boldsymbol{y}}^{(i)} = F_{\overline{b}_i,\phi_i}(\boldsymbol{r}_{\boldsymbol{y}}^{(i)})$.
    Then the prover P sends $(\{\boldsymbol{t}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{a_i; \boldsymbol{s}_{\boldsymbol{v}}^{(i)}; \boldsymbol{t}_{\boldsymbol{z}}^{(i)}; \boldsymbol{s}_{\boldsymbol{w}}^{(i)}; \boldsymbol{t}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell})$
  - If $\mathsf{ch} = 2$, reveal com_1 and com_3.
    For each $i \in [\ell - 1]$, let $\boldsymbol{e}_{\boldsymbol{v}}^{(i)} = \boldsymbol{v}_i^* + \boldsymbol{r}_{\boldsymbol{v}}^{(i)}$;
    For each $i \in [\ell]$, let $c_i = b_i; \widehat{\pi}_i = \pi_i; \widehat{\phi}_i = \phi_i; \boldsymbol{e}_{\boldsymbol{z}}^{(i)} = \boldsymbol{z}_i + \boldsymbol{r}_{\boldsymbol{z}}^{(i)}; \boldsymbol{e}_{\boldsymbol{y}}^{(i)} = \boldsymbol{y}_i + \boldsymbol{r}_{\boldsymbol{y}}^{(i)}$.
    Then the prover P sends $(\{\boldsymbol{e}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{c_i; \widehat{\pi}_i; \widehat{\phi}_i; \boldsymbol{e}_{\boldsymbol{z}}^{(i)}; \boldsymbol{e}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell})$.
  - If $\mathsf{ch} = 3$, reveal com_1 and com_2.
    For each $i \in [\ell - 1]$, let $\boldsymbol{p}_{\boldsymbol{v}}^{(i)} = \boldsymbol{r}_{\boldsymbol{v}}^{(i)}$.
    For each $i \in [\ell]$, let $d_i = b_i; \tilde{\pi}_i = \pi_i; \tilde{\phi}_i = \phi_i; \boldsymbol{p}_{\boldsymbol{z}}^{(i)} = \boldsymbol{r}_{\boldsymbol{z}}^{(i)}; \boldsymbol{p}_{\boldsymbol{y}}^{(i)} = \boldsymbol{r}_{\boldsymbol{y}}^{(i)}$
    Then the prover P sends $(\{\boldsymbol{p}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{d_i; \tilde{\pi}_i; \tilde{\phi}_i; \boldsymbol{p}_{\boldsymbol{z}}^{(i)}; \boldsymbol{p}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell})$.

Figure 5.4: Stern's type zero-knowledge proof system for the accumulated values

- **Output**: The verifier V checks upon ch it chose as follow:
  - If $\mathsf{ch} = 1$:
    For each $i \in [\ell]$, let $\boldsymbol{s}_{\boldsymbol{z}}^{(i)} = \mathsf{ext}(a_i, \boldsymbol{s}_{\boldsymbol{v}}^{(i)})$ and $\boldsymbol{s}_{\boldsymbol{y}}^{(i)} = \mathsf{ext}(a_i, \boldsymbol{s}_{\boldsymbol{w}}^{(i)})$.
    Check that
    $\mathsf{com}_2 = \mathsf{Com}(\{\boldsymbol{t}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{\boldsymbol{t}_{\boldsymbol{z}}^{(i)}; \boldsymbol{t}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell})$
    $\mathsf{com}_3 = \mathsf{Com}(\{\boldsymbol{s}_{\boldsymbol{v}}^{(i)} + \boldsymbol{t}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1}; \{\boldsymbol{s}_{\boldsymbol{z}}^{(i)} + \boldsymbol{t}_{\boldsymbol{z}}^{(i)}; \boldsymbol{s}_{\boldsymbol{y}}^{(i)} + \boldsymbol{t}_{\boldsymbol{y}}^{(i)}\}_{i=1}^{\ell})$.
    And check that $\boldsymbol{s}_{\boldsymbol{v}}^{(i)}, \boldsymbol{w}_{\boldsymbol{v}}^{(i)} \in \mathsf{B}(m, nk)$ for all $i \in [\ell]$.
  - If $\mathsf{ch} = 2$, check that
    $\mathsf{com}_1 = \mathsf{Com}(\{c_i; \widehat{\pi}_i; \widehat{\phi}_i\}_{i=1}^{\ell}; \boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{z}}^{(1)} + \boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{y}}^{(1)} - \boldsymbol{G} \cdot \boldsymbol{u};$
    $\{\boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{z}}^{(i+1)} + \boldsymbol{A}^* \cdot \boldsymbol{e}_{\boldsymbol{y}}^{(i+1)} - \boldsymbol{G}^* \cdot \boldsymbol{e}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1})$
    $\mathsf{com}_3 = \mathsf{Com}(\{\widehat{\pi}_i(\boldsymbol{e}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{F_{c_i, \widehat{\pi}_i}(\boldsymbol{e}_{\boldsymbol{z}}^{(i)}); F_{\bar{c}_i, \widehat{\phi}_i}(\boldsymbol{e}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell})$.
  - If $\mathsf{ch} = 3$, check that
    $\mathsf{com}_1 = \mathsf{Com}(\{d_i; \widetilde{\pi}_i; \widetilde{\phi}_i\}_{i=1}^{\ell}; \boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{z}}^{(1)} + \boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{y}}^{(1)};$
    $\{\boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{z}}^{(i+1)} + \boldsymbol{A}^* \cdot \boldsymbol{p}_{\boldsymbol{y}}^{(i+1)} - \boldsymbol{G}^* \cdot \boldsymbol{p}_{\boldsymbol{v}}^{(i)}\}_{i=1}^{\ell-1})$
    $\mathsf{com}_2 = \mathsf{Com}(\{\widetilde{\pi}_i(\boldsymbol{p}_{\boldsymbol{v}}^{(i)})\}_{i=1}^{\ell-1}; \{F_{d_i, \tilde{\pi}_i}(\boldsymbol{p}_{\boldsymbol{z}}^{(i)}); F_{\bar{d}_i, \tilde{\phi}_i}(\boldsymbol{p}_{\boldsymbol{y}}^{(i)})\}_{i=1}^{\ell})$

  The verifier V outputs 1 if all the checks are passed, otherwise outputs 0.

Figure 5.4: Stern's type zero-knowledge proof system for the accumulated values (cont.)

Chapter 6

Incremental Hash Functions

Hash functions, one of the fundamental cryptographic primitives, map a bit string of arbitrary length to a bit string of fixed length. In that sense, there should be some possibility that a hash function can be transformed into an accumulator, since they both map the large domains to smaller sets. In Section 6.1, we recall some basic properties of hash functions. In Section 6.2, we introduce the definitions of incremental hash functions and two construction paradigms, called *randomize-then-combine* paradigm [BM97] and multiset paradigm [CDv+03, LKMW19]. In Section 6.3, we analyze the incremental hash function LtHash from lattice problems.

6.1 Hash Functions

A hash function is defined by a generator $\mathcal{H}$, which on input a security parameter k outputs the description of a function $h : \{0,1\}^* \mapsto \{0,1\}^k$. A secure cryptographic hash function should satisfy the following properties:

- *Preimage Resistance*: Given some $y \in \{0,1\}^k$, it is infeasible for a probabilistic polynomial-time (PPT) adversary to find a value $x' \in \{0,1\}^*$ such that $h(x') = y$.
- *Second Preimage Resistance*: Given some $x \in \{0,1\}^*$, it is infeasible for a PPT adversary to find a different $x' \in \{0,1\}^*$ such that $h(x) = h(x')$.
- *Collision Resistance*: It is infeasible for any PPT adversary to find two distinct messages $x, x' \in \{0,1\}^*$ such that $h(x) = h(x')$.

Those are three levels of security. If an adversary can do a second preimage attack, they can also do a collison attack – intuitively, if given x an adversary can find x' for which $h(x) = h(x')$, then he can just choose x himself and then run the algorithm for a second preimage attack. Likewise, a hash function that is second preimage resistant is also preimage resistant – if an adversary can invert y and find an x' such that $h(x') = y$, then he can just take some x, compute $y = h(x)$ and invert it to get x'. Since the domain of h is infinite, $x = x'$ happens with negligible probability. We conclude that the three security properties form

a hierarchy where each property implies the one above it, and thus collision resistance is usually used to define security of a hash function.

Collision-resistance property can be defined in the random oracle model as well. Let $h^{(\mathcal{O})}$ be the algorithm h having access to the random oracle $\mathcal{O}$.

Definition 6.1 ($\mathsf{Exp}^{\mathsf{RO}}_{\mathsf{cr}}$)**.** Let λ be the security parameter, given an algorithm h and an efficient adversary $\mathcal{A}$, in the presence of a random oracle RO controlled by the challenger, the experiment $\mathsf{Exp}^{\mathsf{RO}}_{\mathsf{cr}}(h, \mathcal{A})$ for h is defined as follows:

1. The adversary $\mathcal{A}$ submits an integer Q to the challenger representing the maximum number of the unique random oracle queries he will make.

2. The adversary $\mathcal{A}$ makes up to Q unique random oracle queries to the challenger. For each query $i \in [Q]$, the adversary submits $x_i \in \{0,1\}^*$, the challenger responds with $\mathcal{O}(x_i)$.

3. The adversary $\mathcal{A}$ outputs two distinct x, y in the domain of the function h. The output of the experiment is 1 if $h^{(\mathcal{O})}(x) = h^{(\mathcal{O})}(y)$. Otherwise, the output of the experiment is 0.

The advantage of the adversary $\mathcal{A}$ for the function h in the experiment $\mathsf{Exp}^{\mathsf{RO}}_{\mathsf{cr}}$ is defined as

$$\mathsf{Adv}^{\mathsf{RO}}_{\mathsf{cr}}(h, \mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{RO}}_{\mathsf{cr}}(\mathcal{A}) = 1\right].$$

The function h is *collision resistant* in the random oracle model if for all efficient adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{RO}}_{\mathsf{cr}}(h, \mathcal{A})$ is negligible in λ.

6.1.1 Merkle-Damgård Construction

The most common way to construct a hash function is to iterate a compression function on the input message, known as Merkle-Damgård construction [Mer90, Dam90]. In this method, if there exists a collision-resistant compression function with finite input length, it can be used to construct a collision-resistant hash function with arbitrary length inputs. This construction has been used in the design of many hash algorithms such as MD5, SHA-1 and SHA-2.

Given a collision-resistant compression function $f : \{0,1\}^m \mapsto \{0,1\}^k$ where $m - k > 1$. Let $v = m - k - 1$ and the hash function $h : \{0,1\}^* \mapsto \{0,1\}^k$ is defined as follows:

1. The input string x is split into v-bit blocks $x_1, \ldots, x_n$, where x_n is padded with 0s if needed. A block x_{n+1} is appended containing the binary notation of the number of 0s used when padding x_n.

2. A series of m-bit blocks $z_1, \ldots, z_{n+1}$ is defined as $z_1 = 0^k||1||x_1$ and $z_i = f(z_{i-1})||0||x_i$ for $i = 2, \ldots, n+1$.

3. The hash function is finally defined as $h(x) = f(z_{n+1})$.

Now we show that the hash function from this construction is collision-resistant.

Proof. Assume an adversary can find a collision for h, i.e., $x \neq x'$ but $h(x) = h(x')$. Following the construction of h, let $z_1, \ldots, z_{n+1}$ and $z'_1, \ldots, z'_{n'+1}$ be the sequences for the two inputs x and x' respectively. Then we have $f(z_{n+1}) = f(z'_{n+1})$.

Now, there are two cases: (1) $z_{n+1} \neq z'_{n+1}$, which gives a collision for f and it is done. (2) $z_{n+1} = z'_{n'+1}$, which implies that the number of 0s used to padding x_n and x'_n is the same and $f(z_n) = f(z'_{n'})$. Assume without loss of generality that $n \leq n'$. Again, we have two cases: either $z_n \neq z'_{n'}$ leading to a collision for f or $z_n = z'_{n'}$ implying that $x_n = x'_{n'}$ and $f(x_{n-1}) = f(x'_{n'-1})$, which runs into the case analysis repeatedly.

At some stage, there would be a collision for f, otherwise, at a final stage, we could have $z_{n-n'+1} = z'_1$ when we also have $x_{n+1} = x'_{n'+1}, x_n = x'_{n'}, \ldots, x_{n-n'+1} = x'_1$. However, this can never happen: if $n = n'$, this implies $x = x'$ which is a contradiction to the assumption. If $n > n'$, it cannot be the case that $z_{n-n'+1} = z'_1$ since z'_1 has an 1 in the middle (position $k+1$) while $z_{n-n'+1}$ has a 0 in the corresponding position.

Hence, we can conclude that if one can find a collision for h, then he can find a collision for f. □

In the above construction, since $m - k > 1$, there is room for an extra bit marking the start of the hashing process. While in the case $m - k = 1$, there is not. A similar construction can be used to define a function $\bar{h}$, which will be modified to get an actual hash function h. In the same way, we get a sequence of $x_1, \ldots, x_{n+1}$ from the input x. Then, we set $z_1 = 0^k || x_1$ and $z_i = f(z_{i-1}) || x_i$ for $i = 2, \ldots, n+1$. Finally, $\bar{h}$ is defined as $\bar{h} = f(z_{n+1})$. With a suffix-free encoding $E(\cdot)$, the hash function is defined as $h(x) = \bar{h}(E(x))$.

We argue in the same way as before that if one can find a collision for $\bar{h}$, he can find one for f. The argument works almost the same except that in the final stage, we could have $z_{n-n'+1} = z'_1$ which means x' is a suffix of x. However, with the help of suffix-free encoding, there is no pair x, x' such that $E(x')$ is a suffix of $E(x)$. Hence, we can get a collision for f from a collision for h.

6.1.2 Drawbacks of the Iteration Method

The standard hash functions based on Merkle-Damgård construction iteratively use a compression function, and the security is from the cryptanalysis of the compression function. However, the computation is not in parallel and when there are changes applied on the input, the hash value should be computed from the scratch.

6.2 Incremental Hash Functions

The notion of incrementality was proposed by [BGG94]. In the case of hashing, given an incremental hash function h, a message x' is a simple modification of a x which was previously hashed. Instead of re-computing the hash from scratch, $h(x')$ can be derived by updating the old hash value $h(x)$.

Currently, there are two main approaches to construct an incremental collision-free hash function, one named *randomize-then-combine* paradigm [BM97] and another paradigm for multiset [CDv+03, LKMW19]. Under both paradigms, there are different types of constructions: AdHash, MuHash and LtHash using additions, modular multiplications and lattice-based functions respectively. We will show both construction paradigms in this section, and take LtHash as an example for each style.

6.2.1 Randomize-then-combine Paradigm

An incremental hash function was first proposed in [CvP92, BGG94] where the function is based on discrete exponentiation in a group of prime order, and uses one modular exponentiation for each message block to hash. That construction is expensive, i.e., to hash an n-block message, $n \cdot k$ bits in total, it requires n exponentiations modulo a $O(k)$-bit prime. Compared with MD5, fixing $k = 512$, n exponentiations is more expensive than MD5 on $512n$ bits. The *randomize-then-combine* paradigm [BM97] reduces the cost.

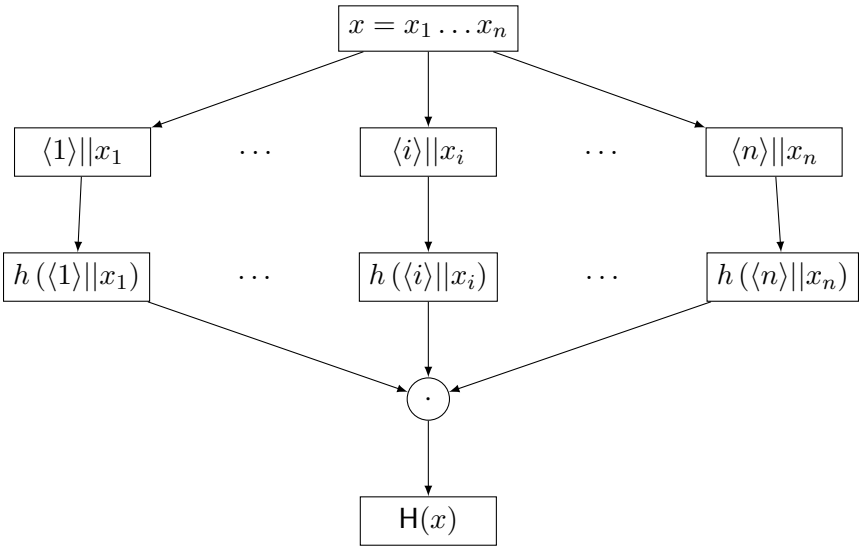

Figure 6.1: Random-then-combine paradigm for hashing message $x = x_1 \dots x_n$ where $\odot$ is some operation.

The paradigm is shown in Figure 6.1. It takes as input a message x, which is viewed as a sequence of blocks, $x = x_1 \dots x_n$, each block of b bits. First, each block x_i is processed via a *randomizing function* h to yield a result $h(\langle i \rangle || x_i)$. Then, the results are *combined* with some operation $\odot$. Formally,

Definition 6.2 (Incremental Hash Function [BM97])**.** Let b be the fixed block size, N be the upper bound of the number of blocks, $l = \lg N + b$, $(G, \odot)$ be a commutative group and h be a function $h : \{0,1\}^l \mapsto G$. A hash function $\mathsf{H} : \{0,1\}^{b \cdot N} \mapsto G$ defined as

$$\mathsf{H}(x_1 \dots x_n) = \bigodot_{i=1}^{n} h(\langle i \rangle || x_i)$$

Function	Based Group	$\mathsf{H}(x_1 \dots x_n)$	Computational Hard Problem
AdHash	$\mathbb{Z}_M$	$\sum_{i=1}^{n} h(\langle i \rangle \| \| x_i) \bmod M$	Standard modular knapsack
MuHash	G	$\prod_{i=1}^{n} h(\langle i \rangle \| \| x_i)$	Discrete logarithm problem
LtHash	$\mathbb{Z}_q^k$	$\sum_{i=1}^{n} h(\langle i \rangle \| \| x_i) \bmod q$	Short integer solution

Table 6.1: Secure incremental hash functions from random-then-combine paradigm

is a collision-resistant incremental hash function if it satisfies:

- *Collision-resistance:* It is computationally infeasible to find two distinct inputs $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ such that $\mathsf{H}(x_1 \dots x_n) = \mathsf{H}(y_1 \dots y_m)$.
- *Incrementality:* When x_i is modified as x_i', one can simply re-compute the hash value given by $y = \mathsf{H}(x_1 \dots x_i \dots x_n)$ as $\mathsf{H}(x_1 \dots x_i' \dots x_n) = y \odot h(\langle i \rangle \| \| x_i)^{-1} \odot h(\langle i \rangle \| \| x_i')$.

In the construction, the *randomizing function* h should be collision-free and applied to fixed length inputs, and thus it is treated as a random oracle while in practice is derived from some standard hash function like SHA-256. The combining operation $\odot$ is a group operation depending on the based group used for H.

There are three types of secure H functions, summarized in Table 6.1.

AdHash function

AdHash is derived by setting the *combining operation* to addition modulo a large integer M and the *randomizing function* to a hash function $h : \{0,1\}^l \mapsto \mathbb{Z}_M$.

MuHash function

MuHash is derived by setting the *combining operation* to multiplication in a group where the discrete logarithm problem is hard, e.g. $G = \mathbb{Z}_q^*$, and the *randomizing function* to a hash function $h : \{0,1\}^l \mapsto G$.

LtHash function

LtHash is derived by setting the *combining operation* to component-wise addition modulo an integer p, i.e., the output is a k-entry column vector over $\mathbb{Z}_q$ and the *randomizing function* to a hash function $h : \{0,1\}^l \mapsto \mathbb{Z}_q^k$. The detailed construction is discussed in Section 6.3.

There is another candidate for an incremental hash function, XHash, where the *combining operation* is set to bitwise XOR and the security does not depend on any number theoretical problem. However, this is not collision-free and an attack using Gaussian elimination can break the security.

6.2.2 Set Incremental Hash Functions

In Figure 6.1, the incremental hash function depends on the order of the inputs. While in set incremental hash functions, the order is not important. Similar as

Definition 6.2, an underlying hash function and a group operation are used to construct a set incremental hash function H. Formally,

Definition 6.3 (Set Incremental Hash Function [CDv+03, LKMW19])**.** For an input set $X = \{x_1, \ldots, x_n\} \in \mathcal{P}(\{0,1\}^*)$, a commutative group G with operation $\odot$, and an underlying hash function $h : \{0,1\}^* \mapsto G$ (modeled as a random oracle, or as a standard hash function in practice), the set incremental hash function $\mathsf{H} : \mathcal{P}(\{0,1\}^*) \mapsto G$ is defined as

$$\mathsf{H}(X) = \bigodot_{x_i \in X} h(x_i)$$

which satisfies the following properties:

- *Collision-resistance:* It is computationally infeasible to find two input sets $S, T \in \mathcal{P}(\{0,1\}^*)$ such that $\mathsf{H}(S) = \mathsf{H}(T)$.
- *Set Incrementality:* For any two disjoint sets $S, T \in \mathcal{P}(\{0,1\}^*)$, we have $\mathsf{H}(S \cup T) = \mathsf{H}(S) \odot \mathsf{H}(T)$. This implies incrementality from element addition or deletion.

6.3 Hashing from Lattice

Ajtai introduced an one-way function based on the average-case hardness problem, *short integer solution* [Ajt96]. This method can be used to construct families of collision-free hash functions [GGH96]. This hash function is indeed incremental, but not practical. Following the paradigms shown in 6.2.1 and 6.2.2, more practical (set) incremental hash functions from lattice have been described [BM97, LKMW19].

6.3.1 Ajtai-GGH Function

Let $\boldsymbol{M}$ be a random k-by-n matrix with entries in $\mathbb{Z}_q$ and $\boldsymbol{x}$ be an n vector with entries in $\{0,1\}$. Let $k\log(q) < n \leq \frac{p}{2k^4}$, the Ajtai-GGH function is

$$H_{\boldsymbol{M},q}(\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x} \bmod q \tag{6.1}$$

The function is compressing, incremental, one-way and collision-free, more specifically,

- *compression.* Given the restriction on security parameters, $k\log(q) < n \leq q/(2k^4)$, the input n-vector $\boldsymbol{x}$ with each entry in $\{0,1\}$, n-bit long, is mapped into a k-vector over $\mathbb{Z}_q$, $k\log(q)$-bit long, and thus the function is compressing.
- *incrementality.* Let $\boldsymbol{M}_i$ denote the i-th column of $\boldsymbol{M}$ for $i = 1, \ldots, n$, and it is a k-vector over $\mathbb{Z}_q$. Let $\boldsymbol{x} = (x_1 \ldots x_n)$ with $x_i \in \{0,1\}$ for $i = 1, \ldots, n$. The function can be written as

$$H_{\boldsymbol{M},q}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \boldsymbol{M}_i \mod q \tag{6.2}$$

Suppose bit x_i changes to x_i' and we have the old hash value y, then the new hash value is $y + (x_i' - x_i)\boldsymbol{M}_i \mod p$, which takes running k additions modulo p whose time is independent of the length of x.

- *collision-resistance.* Suppose two distinct messages $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ and $\boldsymbol{y} = \{y_1, \ldots, y_n\}$ result in the same hash value. We have $\boldsymbol{M}_1 x_1 + \cdots + \boldsymbol{M}_n x_n = \boldsymbol{M}_1 y_1 + \cdots + \boldsymbol{M}_n y_n \mod q$, which means $\boldsymbol{M}_1(x_1 - y_1) + \cdots + \boldsymbol{M}_m(x_n - y_n) = 0 \mod q$. As it is hard to solve the SIS problem, we cannot find a pair of $\boldsymbol{x}$ and $\boldsymbol{y}$ that satisfy this equation where $x_i - y_i \in \{-1, 0, +1\}$ for all $i \in [n]$.

Remark. Even though Ajtai-GGH function has the properties for incremental hash function, it is not practical. One drawback of this function is that the description, $\boldsymbol{M}, q$, is large, $(nk+1)\log(q)$ bits, and it depends on the number of bits to be hashed. In this way, we need to set a limitation for the length of the inputs to be hashed.

There are two options to overcome the problem. One is to specify $\boldsymbol{M}$ from an ideal hash function, i.e., given a hash $h : [k] \times [n] \mapsto \mathbb{Z}_q$, $\boldsymbol{M}[i, j] = h(i, j)$. Another one is to use iterative methods shown in Section 6.1.1 but it will lose incremental property.

6.3.2 Incremental Hashing from Lattice

Defining $\boldsymbol{M}$ via an ideal hash function can be a solution to reduce the size of hash description. But following the *randomize-then-combine* paradigm, an efficient lattice-based incremental hash function can be derived, with small key size and no limitation on the input length. The construction of LtHash is an application of the paradigm where the group is set to $\mathbb{Z}_q^k$.

Let $\boldsymbol{x} = (x_1 \ldots x_n)$ be the data to be hashed, b be the block size. Assume that all messages we need to hash have length at most N and let $l = b + \log(N)$. Let $h : \{0,1\}^l \mapsto \mathbb{Z}_q^k$ be a hash function, whose output is a k-entry column vector over $\mathbb{Z}_q$. The hash function now is

$$\mathsf{LtHash}_q^h(x_1 \ldots x_n) = \sum_{i=1}^{n} h(\langle i \rangle \,||x_i) \bmod q \tag{6.3}$$

In this construction, each application of h yields a column vector, and these are added, component-wise modulo q, to get a final column vector which is the hash value.

Theorem 6.1. *The incremental hash function in Equation 6.3 is collision-resistant from the hardness of matrix-kernel problem (*SIS *problem with* $\beta = 1$*).*

Proof. We are given a collision-finder C, which takes q and an oracle for h, and outputs a pair of distinct strings, $\boldsymbol{x} = (x_1 \ldots x_n)$ and $\boldsymbol{y} = (y_1 \ldots y_m)$, such that $\mathsf{LtHash}_q^h(\boldsymbol{x}) = \mathsf{LtHash}_q^h(\boldsymbol{y})$. We can construct an algorithm K that solves the matrix-kernel problem efficiently, which takes as input q and a list of values $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_p$ selected uniformly at random in $\mathbb{Z}_q^k$. K runs C on input q, responding to the distinct oracle queries with the values $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_p$ in order.

Let Q_i denote the i-th oracle query of C, which will be answered by a_i, so that $h(Q_i) = \boldsymbol{a}_i$ since the answers are uniformly and independently distributed over G and $h : \{0,1\}^l \mapsto G$ is a random function. Finally, C outputs two strings $\boldsymbol{x} = (x_1 \dots x_n)$ and $\boldsymbol{y} = (y_1 \dots y_m)$ such that $x \neq y$ but $\mathsf{LtHash}(\boldsymbol{x}) = \mathsf{LtHash}(\boldsymbol{y})$, which means

$$\sum_{i=1}^{n} h(\langle i \rangle \,||x_i) = \sum_{i=1}^{m} h(\langle i \rangle \,||y_i) \bmod q \tag{6.4}$$

Let $x'_i = \langle i \rangle || x_i$ for $i = 1, \dots, n$ and $y'_i = \langle i \rangle || y_i$ for $i = 1, \dots, m$. We assume that $x'_1, \dots, x'_n, y'_1, \dots, y'_m \in Q$. Let $f_x(i)$ be the unique value $j \in [p]$ that $x'_i = Q_j$ and $f_y(i)$ be the unique value $j \in [p]$ that $y'_i = Q_j$. Then, let $I = \{f_{\boldsymbol{x}}(i) : i = 1, \dots, n\}$ and $J = \{f_{\boldsymbol{y}}(i) : i = 1, \dots, m\}$ be the indices of queries corresponding to $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively. Equation 6.4 can be rewritten as

$$\sum_{i \in I} \boldsymbol{a}_i = \sum_{j \in J} \boldsymbol{a}_j \mod p \tag{6.5}$$

Since $\boldsymbol{x} \neq \boldsymbol{y}$, it holds that $I \neq J$. For $i = 1, \dots, p$, let $w_i = \begin{cases} -1 \text{ if } i \in J - I \\ 0 \text{ if } i \in I \cap J \\ +1 \text{ if } i \in I - J \end{cases}$. Then since $I \neq J$, not all $w_1, \dots, w_p$ are 0, and Equation 6.5 implies $w_1 \boldsymbol{a}_1 + \dots + w_p \boldsymbol{a}_p = 0 \bmod q$. The probability that we find a solution to the SIS problem is the same with which C outputs a collision. □

6.3.3 Set Incremental Hashing from Lattice

Let N, d be positive integers, $h : \{0,1\}^* \mapsto \{0,1\}^{Nd}$, where the output can be represented as a vector of N blocks each of length d bits. For an input $x \in \{0,1\}^*$, the output is written as $\boldsymbol{h}(x) = ([h(x)]_1, \dots, [h(x)]_N)$. Using $p = 2^d$, the hash function $\mathsf{LtHash}_{N,d} : \mathcal{P}(\{0,1\}^*) \mapsto \mathbb{Z}_q^N$ is defined as

$$\mathsf{LtHash}_{N,d}(\{x_1, \dots, x_n\}) = \sum_{i=1}^{n} \boldsymbol{h}(x_i) \bmod q$$

where the sum operation is component-wise addition modulo q.

Theorem 6.2. *The set incremental hash function is collision-resistant from the hardness of* SIS *problem.*

Proof. To prove the collision resistance property in the random oracle model from short integer solution, we need an efficient simulator Sim, which acts as the challenger in $\mathsf{Exp}^{\mathsf{RO}}_{\mathsf{cr}}$ and the adversary in $\mathsf{Exp}_{\mathsf{SIS}}$. In this simulation, Sim will respond to the random oracle queries submitted by $\mathcal{A}$ and keep track of the responses as mappings from unique queries to random oracle outputs in a key-value store $\mathcal{M}$. The simulator works as follows:

1. The adversary $\mathcal{A}$ submits an integer Q to the simulator Sim to represent the maximum number of random oracle queries he will make.

2. The simulator Sim runs the experiment $\mathsf{Exp}_{\mathsf{SIS}}$ with parameters $(n, Q, 2^d, 1)$, and the challenger samples $\boldsymbol{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{n\times Q}$ and sends to Sim.

3. The simulator Sim views the matrix $\boldsymbol{A}$ as a Q-vector $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_p)$. When the adversary $\mathcal{A}$ submits the i-th random oracle query $q_i \in \{0,1\}^*$, the simulator Sim will respond with $\boldsymbol{a}_i$ and store the key-value mapping $(q_i \mapsto \boldsymbol{a}_i)$ in the store $\mathcal{M}$.

4. The adversary $\mathcal{A}$ outputs two distinct non-empty sets $S = \{s_1, \ldots, s_{|S|}\}$ and $T = \{t_1, \ldots, t_{|T|}\}$ and sends the two sets to the simulator Sim.

5. The simulator Sim computes $S \cup T$. If there is no matching key entries in $\mathcal{M}$ for at least one element in $S \cup T$, Sim will abort the experiment by outputting an all-one vector $\boldsymbol{x} = 1^Q \in \mathbb{Z}^Q$ to the challenger. Otherwise, Sim will construct the vector $\boldsymbol{x} \in \mathbb{Z}^Q$ as follows, for each $i \in [Q]$:

$$\boldsymbol{x}_i = \begin{cases} 1, & \text{if } \exists q \in S \backslash T \text{ s.t. } (q \mapsto \boldsymbol{a}_i) \text{ is stored in } \mathcal{M} \\ -1, & \text{if } \exists q \in T \backslash S \text{ s.t. } (q \mapsto \boldsymbol{a}_i) \text{ is stored in } \mathcal{M} \\ 0, & \text{otherwise} \end{cases}$$

   and send the vector $\boldsymbol{x}$ to the challenger.

First, we show the correctness of the Sim construction, i.e., it is indeed a simulation of the challenger in $\mathsf{Exp}_{\mathsf{cr}}^{\mathsf{RO}}$ and the adversary in $\mathsf{Exp}_{\mathsf{SIS}}$.

As the challenger in $\mathsf{Exp}_{\mathsf{cr}}^{\mathsf{RO}}$, the responses from the simulator Sim to the adversary $\mathcal{A}$ are columns of the matrix $\boldsymbol{A}$ obtained from the challenger in $\mathsf{Exp}_{\mathsf{SIS}}$, which is sampled uniformly at random. Hence, the distribution of the responses by the simulator Sim is identical to the distribution of the outputs made by the challenger in the experiment in the random oracle model.

As the adversary in $\mathsf{Exp}_{\mathsf{SIS}}$, we show that the output of the simulator Sim is always a non-zero vector $\boldsymbol{x} \in \mathbb{Z}^Q$ with $||\boldsymbol{x}||_\infty \leq 1$. When Sim aborts, it outputs $\boldsymbol{x} = 1^Q$ which is indeed an non-zero vector satisfying the ℓ_∞-norm restriction. Otherwise, there are mappings in $\mathcal{M}$ for elements in $S \cup T$ as entries and $S \neq T$. In this case, the vector $\boldsymbol{x}_i$ can be zero only if there exists an element $q \in S \cap T$ such that $(q \mapsto \boldsymbol{a}_i)$ is stored in $\mathcal{M}$. Moreover, if this holds for all $i \in [Q]$, the vector $\boldsymbol{a}$ can be all-zero, implying that $S = T$, which is a contradiction. This means that the vector $\boldsymbol{x}$ output by the simulator Sim is always non-zero and satisfies the ℓ_∞-norm restriction.

Now we show that if the adversary $\mathcal{A}$ can win in the experiment $\mathsf{Exp}_{\mathsf{cr}}^{\mathsf{RO}}$, Sim as the adversary in $\mathsf{Exp}_{\mathsf{SIS}}$ can win as well.

When the simulator Sim aborts, it means the adversary $\mathcal{A}$ outputs two sets S, T where there exists one element in $S \cup T$ not submitted in the random queries. Since the output space is of nd bits and the output is independent and uniformly random from the input, the probability of Sim aborting when $\mathsf{LtHash}_{n,d}(S) = \mathsf{LtHash}_{n,d}(T)$ is $1/2^{nd}$.

Then, consider that the simulator Sim never aborts. Each element in $S \cup T$ corresponds to an entry in $\mathcal{M}$. If $\mathsf{Exp}_{\mathsf{cr}}^{\mathsf{RO}}(\mathcal{A}) = 1$, then $\mathsf{LtHash}_{n,d}(S) -$

$\mathsf{LtHash}_{n,d}(T) = \mathbf{0}$, which means

$$\sum_{i=1}^{|S|} \mathcal{M}(s_i) - \sum_{i=1}^{|T|} \mathcal{M}(t_i) = \mathbf{0} \bmod q$$

equivalent to $\boldsymbol{A}\boldsymbol{x} = \mathbf{0} \bmod q$ and $\boldsymbol{x}$ is defined in Sim. Then we can show that $\mathsf{Exp}_{\mathsf{SIS}}^{(n,Q,2^d,1)} = 1$.

Formally, for all efficient adversaries $\mathcal{A}$ that make up to Q unique random oracle queries, we have that

$$\mathsf{Adv}_{\mathsf{cr}}^{\mathsf{RO}}(\mathsf{LtHash}_{n,d}, \mathcal{A}) \leq \mathsf{Adv}_{\mathsf{SIS}}^{(n,Q,2^d,1)}(\mathsf{Sim}) + \frac{1}{2^{nd}}$$

□

Chapter 7

Attempts and Failures

In this chapter, we show our attempts and failures about the constructions and applications of lattice-based accumulators. In Section 7.1, we attempt to apply Stern's protocol for the compact accumulator with trapdoor [JS15] to construct a zero-knowledge proof system. In Section 7.2, we discuss the possibilities to build a PSI protocol from the current lattice-based accumulator schemes. In Section 7.3, we try to transform the lattice-based incremental hash function into an accumulator. In Section 7.4, we follow the laconic PSI paradigm based on accumulators to construct a PSI protocol from incremental hashing.

7.1 Zero-knowledge Proof for Accumulator [JS15]

We show the construction of a compact accumulator from lattice [JS15] in Section 5.1. It is the first lattice-based accumulator scheme and it is secure from the hardness assumption of the SIS problem which is implied by the worst-case problems. One interesting problem that work left is to build a zero-knowledge proof system for the accumulated values.

We try to apply Stern's protocol [Ste94a] in Figure 3.2 here. In the protocol, on input $(\boldsymbol{A}, v_S)$, the prover P needs to convince the verifier V that P owns a value-witness pair $(\boldsymbol{B}, w)$ which is accepted by the verification algorithm. Then, the relation R can be defined as

$$R = \{(\boldsymbol{A}, v_S) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times m'}; \boldsymbol{B} \in \mathbb{Z}_q^{n\times m'}; w \in \mathbb{Z}^{(m+m')} : \mathsf{Ver}(\boldsymbol{A}, \boldsymbol{B}, w, v_S) = 1\}$$

To be specific, $\mathsf{Ver}(\boldsymbol{A}, \boldsymbol{B}, w, v_S) = 1$ implies $w \in \Lambda^{\perp}(\boldsymbol{F_B})$ where and $0 < \|w\| \leq s\sqrt{m+m'}$.

Recall that in Stern's protocol, we have $\boldsymbol{Ax} \bmod q = \boldsymbol{y}$ as the statement and the three commitments are made on the (permutated) $\boldsymbol{Ar}$, $\boldsymbol{r}$ and $\boldsymbol{x} + \boldsymbol{r}$ where $\boldsymbol{r}$ is randomly sampled. The public matrix $\boldsymbol{A}$ in the statement is essential to check the commitments from the verifier V's side.

Now, back to the case of this compact accumulator, even though we still have the $\boldsymbol{A}$ as an public input to the accumulator scheme, the verification algorithm Ver requires more than $\boldsymbol{A}$, i.e., $\boldsymbol{F_B}$. The matrix $\boldsymbol{F_B}$ is an extension of $\boldsymbol{A}$ and it is computed as

$$\boldsymbol{F_B} = [\boldsymbol{A}||(v_S - \boldsymbol{B})].$$

If we want to prove that $w \in \Lambda^{\perp}(\boldsymbol{F}_{\boldsymbol{B}})$ in Stern's style, it seems that we have to make $\boldsymbol{F}_{\boldsymbol{B}}$ something public. However, $\boldsymbol{F}_{\boldsymbol{B}}$ depends on the value the prover P owns and for each possible $\boldsymbol{B}$, $\boldsymbol{F}_{\boldsymbol{B}}$ is totally different.

Hence, the Stern's protocol cannot be applied to build a zero-knowledge proof system for this accumulator. Moreover, this statement cannot be used for any zero-knowledge proof system since $\boldsymbol{F}_{\boldsymbol{B}}$ may leak information of $\boldsymbol{B}$. The statement should be extended to some statements with public inputs independent of $\boldsymbol{B}$. However, we have not got any solution to this.

7.2 PSI from Lattice-based Accumulators [JS15, LLNW16]

A succinct PSI protocol constructed from RSA-accumulators with trapdoor [ADT11] fits the paradigm shown in Figure 4.1 except that the sender has the trapdoor.

Now, we briefly discuss the possibility to construct a PSI protocol from the current two lattice-based accumulators. Unfortunately, there is no method to do that since the constructions of the two accumulators are more complex than the ideal accumulator. Recall that the ideal accumulator should satisfy the following properties and both should be done by the server – the sender does not create a witness itself but sends the components for the verification algorithm:

- The witness w for an element y is not a function of y, instead, it is generated from the set of X except the element x, i.e., $w = \mathsf{Wit}(X_{-y})$.
- The verification function can be described as that if $\mathsf{Ver}(\mathsf{Acc}(X), x, w) = 1$, there exist two functions ψ, ϕ such that $\phi\left(\psi(x), X_{-y}\right) = \mathsf{Acc}(X)$ and the functions output elements in a group where it holds that $\phi\left(\psi(y)^{\beta}, X\right) = \phi(\psi(y), X)^{\beta}$ for all y, X and scalar β.

Due to the basis delegation technique and tree construction, neither of those accumulators can achieve those properties.

7.3 Trapdoorless Accumulator from Incremental Hashing [LKMW19]

We show the construction of set incremental hash functions in Section 6.2.2 and the lattice-based one in Section 6.3.3. It seems an plausible candidate to build an accumulator since it is capable to map a large set into a small representation, and the hash function is incremental – the accumulator built on this can be dynamic. Now, we give the insecure construction of an accumulator from lattice-based accumulators, shown in Figure 7.1, which fit the properties mentioned in Section 4.2.

Here, the accumulation algorithm is to hash a set using the set incremental hash function from lattices. The witness for an element is computed as the hash of the set except that element. To verify the witness is valid, the underlying hash function applied to the element with component-wise addition to the witness should equal the accumulator's value.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input security parameter 1^λ, it generates the hash description for $\mathsf{LtHash}_{N,d} : \mathcal{P}(\{0,1\}^*) \mapsto \mathbb{Z}_q^N$.

  Return $\mathsf{pp} = (q, h)$.

- $\mathsf{Acc}(\mathsf{pp}, X) \to v_X$:

  Return $v_X = \sum_{x_i \in X} \boldsymbol{h}(x_i) \bmod q$ where the sum operation is component-wise addition modulo q. Note that the output of hash function h is represented as a vector of N blocks each of length d bits, written as $\boldsymbol{h}(x) = ([h(x)]_1, \ldots, [h(x)]_N)$.

- $\mathsf{Wit}(\mathsf{pp}, X, x_i) \to w$:

  Return $w = \sum_{x_j \in X \setminus \{x_i\}} \boldsymbol{h}(x_j) \bmod q$.

- $\mathsf{Ver}(\mathsf{pp}, v_X, x, w) \to \texttt{accept}/\texttt{reject}$:

  If $v_X = \boldsymbol{h}(x) + w \bmod q$, then return `accept`, otherwise, return `reject`.

Figure 7.1: Lattice-based Accumulator from Incremental Hashing

It is straightforward to see that the accumulator scheme is correct – an honestly produced witness for the element in the set will always be accepted by the verification algorithm.

To see that the accumulator scheme is collision-resistant, we try to show how to relate it to the collision-resistance property for the incremental hash functions from lattices.

Attempt. *The accumulator construction of Figure 7.1 seems collision-resistant under the hardness of* SIVP_γ *problem.*

Proof. Let $\mathcal{A}$ be a PPT adversary that can break the collision-resistance property of the accumulator. He can output a set $X = \{x_1, \ldots, x_n\}$, a value $y \notin X$ and a witness w^* such that $v_X = \boldsymbol{h}(y) + w^* \bmod q$.

Note that since w^* is a valid witness, it holds that $w^* = \sum_{z_j \in Z} \boldsymbol{h}(z_j) \bmod q$ for some set Z such that $\forall x_i \in X : \sum_{x_k \in X \setminus x_i} \boldsymbol{h}(x_k) \neq \sum_{z_j \in Z} \boldsymbol{h}(z_j) \bmod q$ since $y \notin X$. Then, we can write the verification as

$$\boldsymbol{h}(x_1) + \boldsymbol{h}(x_2) + \ldots \boldsymbol{h}(x_n) = \boldsymbol{h}(y) + \boldsymbol{h}(z_1) + \ldots + \boldsymbol{h}(z_m) \bmod q.$$

Then it follows the collision-resistance property of the incremental hash function from lattices, which follows the hardness of SIVP_γ problem. □

However, the proof is not correct and this accumulator scheme is indeed insecure. It is pretty easy to fabricate a witness for some element not in the set:

Given a value for the accumulator, say v_X for the set X, the adversary can forge a witness for an element $y \notin Y$ as $w = v_X - \boldsymbol{h}(y) \bmod q$, which is not hard to

compute. Then given (y, w) as the value-witness pair, the verification algorithm Ver is more than happy to accept since it always holds that $w + \boldsymbol{h}(y) = v_X \bmod q$.

We try to add some other primitives like public key encryption with equality test [DRS+22], but it makes no differences about the prevention of brute force attack since the linear computation is not hard to do. The only way we can come up with to solve this problem is to raise the output of hash function as exponents – like RSA or pairing. It is not what we aim for, and thus we still leave it as a problem.

7.4 Laconic PSI from Incremental Hashing [LKMW19]

From the "accumulator" construction in Figure 7.1, even though it is not secure, we still try to follow the paradigm shown in Figure 4.1 to construct a laconic PSI protocol and see whether it can be a candidate.

The protocol from this "accumulator" is shown in Figure 7.2. The protocol is correct from the correctness of the accumulator scheme. But it is not secure. Even though we introduce some randomizers on both sides, we still cannot prevent the receiver from knowing the sender's set – The receiver can do a brute force attack, i.e., the randomizers can be cancelled and he does not compute $\sum_{i\in[m]\setminus\{k\}} \boldsymbol{h}(x_i)$ upon receiving from the sender, instead, he computes $\sum_{i\in[m]} \boldsymbol{h}(x_i) - \boldsymbol{h}(z)$ for whatever z he wants to match. Once the check is passed, he knows that z is in the sender's set. It is indeed the insecurity of the accumulator.

Recall the two laconic PSI from the RSA-based and pairing-based accumulators, there are some hardness assumptions to prevent the brute force attacks, ϕ-hiding and B-SBDDH assumption respectively. Again, however, here we have no helpful assumptions to rely on due to the linear computation.

7.5 Other Attempts

To study some state-of-the-art zero-knowledge proofs, we focus on *syndrome decoding in the head* [FJR22b]. It follows the *MPC-in-the-Head* (MPCitH) paradigm [IKOS07] where a zero-knowledge proof is built from secure multi-party computation protocols. They prove the small Hamming weight of the syndrome decoding solutions by proving some relations between specific polynomials with the help of multi-party product verification protocols [BN20, KZ22]. This protocol is 5-round and achieves a soundness error around $1/N$, much lower than the $2/3$ of the Stern's type protocols. However, we haven't found any opportunities to apply similar techniques to the proof system for the lattice-based accumulator.

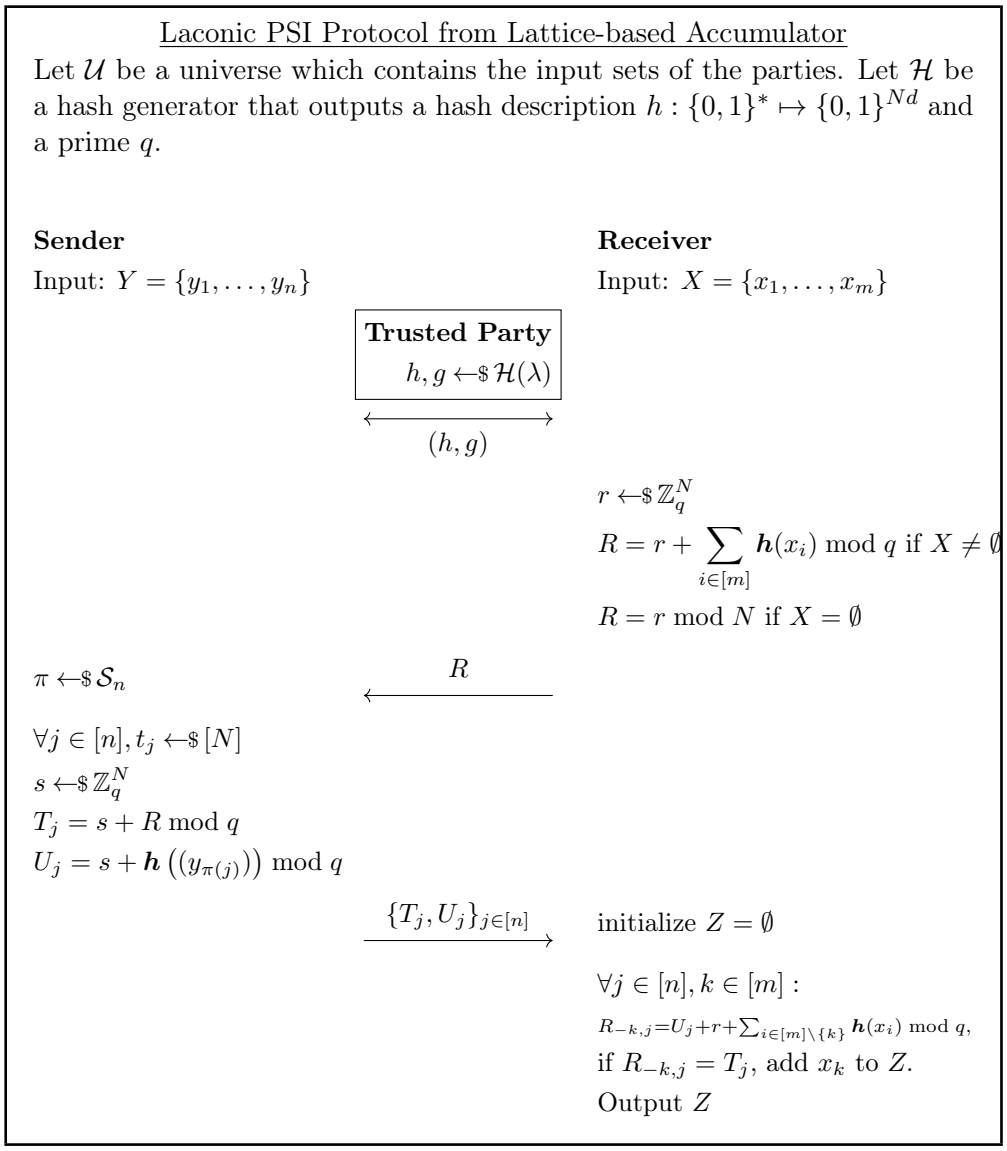

Figure 7.2: The insecure laconic PSI protocol based on lattice-based accumulator

Bibliography

[ABC+12] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 1–20. Springer, Heidelberg, March 2012.

[ABD+21] Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In *Theory of Cryptography Conference*, pages 94–125. Springer, 2021.

[AC20] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Heidelberg, August 2020.

[ACBH13] Sidi Mohamed El Yousfi Alaoui, Pierre-Louis Cayrel, Rachid El Bansarkhani, and Gerhard Hoffmann. Code-based identification and signature schemes in software. In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar R. Weippl, and Lida Xu, editors, *Security Engineering and Intelligence Informatics - CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings*, volume 8128 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2013.

[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*, volume 6571 of *LNCS*, pages 156–173. Springer, Heidelberg, March 2011.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[ALOS22] Diego Aranha, Chuanwei Lin, Claudio Orlandi, and Mark Simkin. Laconic private set-intersection from pairings. Cryptology ePrint Archive, Paper 2022/529, 2022. https://eprint.iacr.org/2022/529.

[ATSM09] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 295–308. Springer, Heidelberg, April 2009.

[AWSM07] Man Ho Au, Qianhong Wu, Willy Susilo, and Yi Mu. Compact e-cash from bounded accumulator. In Masayuki Abe, editor, *CT-RSA 2007*, volume 4377 of *LNCS*, pages 178–195. Springer, Heidelberg, February 2007.

[BCD+17] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. Accumulators with applications to anonymity-preserving revocation. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, pages 301–315. IEEE, 2017.

[BCY20] Foteini Baldimtsi, Ran Canetti, and Sophia Yakoubov. Universally composable accumulators. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 638–666. Springer, Heidelberg, February 2020.

[Bd94] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 274–285. Springer, Heidelberg, May 1994.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993.

[BGG94] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 216–233. Springer, Heidelberg, August 1994.

[BGLS19] Shi Bai, Steven D. Galbraith, Liangze Li, and Daniel Sheffield. Improved combinatorial algorithms for the inhomogeneous short integer solution problem. *Journal of Cryptology*, 32(1):35–83, January 2019.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

[BM97] Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 163–192. Springer, Heidelberg, May 1997.

[BN20] Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526. Springer, Heidelberg, May 2020.

[BP97] Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494. Springer, Heidelberg, May 1997.

[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, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Heidelberg, August 2017.

[CDv+03] Dwaine E. Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend, and G. Edward Suh. Incremental multiset hash functions and their application to memory integrity checking. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 188–207. Springer, Heidelberg, November / December 2003.

[CF13] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Heidelberg, February / March 2013.

[CG10] Sébastien Canard and Aline Gouget. Multiple denominations in e-cash with compact transaction data. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 82–97. Springer, Heidelberg, January 2010.

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, Heidelberg, May / June 2006.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012.

[CJ10] Sébastien Canard and Amandine Jambert. On extended sanitizable signature schemes. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *LNCS*, pages 179–194. Springer, Heidelberg, March 2010.

[CKS09] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer, Heidelberg, March 2009.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2011.

[CP91] Paul Camion and Jacques Patarin. The Knapsack hash function proposed at Crypto'89 can be broken. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 39–53. Springer, Heidelberg, April 1991.

[CvP92] David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 470–484. Springer, Heidelberg, August 1992.

[Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, August 1990.

[Dam02] Ivan Damgård. On σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, page 84, 2002.

[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*, pages 789–800. ACM Press, November 2013.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626. Springer, Heidelberg, May 2004.

[DRS+22] Dung Hoang Duong, Partha Sarathi Roy, Willy Susilo, Kazuhide Fukushima, Shinsaku Kiyomoto, and Arnaud Sipasseuth. Chosen-ciphertext lattice-based public key encryption with equality test in standard model. *Theor. Comput. Sci.*, 905:31–53, 2022.

[DT08] Ivan Damgård and Nikos Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538, 2008. https://eprint.iacr.org/2008/538.

[FJR21] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature. *IACR Cryptol. ePrint Arch.*, page 1576, 2021.

[FJR22a] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. *IACR Cryptol. ePrint Arch.*, page 188, 2022.

[FJR22b] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. Cryptology ePrint Archive, Paper 2022/188, 2022. https://eprint.iacr.org/2022/188.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[GG07] Philippe Gaborit and Marc Girault. Lightweight code-based identification and signature. In *IEEE International Symposium on Information Theory, ISIT 2007, Nice, France, June 24-29, 2007*, pages 191–195. IEEE, 2007.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. *Electron. Colloquium Comput. Complex.*, (42), 1996.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.

[GPS22] Shay Gueron, Edoardo Persichetti, and Paolo Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptogr.*, 6(1):5, 2022.

[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 ACM STOC*, pages 197–206. ACM Press, May 2008.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

[JS15] Mahabir Prasad Jhanwar and Reihaneh Safavi-Naini. Compact accumulator using lattices. In Rajat Subhra Chakraborty, Peter Schwabe, and Jon A. Solworth, editors, *Security, Privacy, and Applied Cryptography Engineering - 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings*, volume 9354 of *Lecture Notes in Computer Science*, pages 347–358. Springer, 2015.

[KB21] Ioanna Karantaidou and Foteini Baldimtsi. Efficient constructions of pairing based accumulators. In Ralf Küsters and Dave Naumann, editors, *CSF 2021 Computer Security Foundations Symposium*, pages 1–16. IEEE Computer Society Press, 2021.

[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*, pages 818–829. ACM Press, October 2016.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 372–389. Springer, Heidelberg, December 2008.

[KZ22] Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Paper 2022/588, 2022. https://eprint.iacr.org/2022/588.

[LKMW19] Kevin Lewi, Wonho Kim, Ilya Maykov, and Stephen Weis. Securing update propagation with homomorphic hashing. Cryptology ePrint Archive, Report 2019/227, 2019. https://eprint.iacr.org/2019/227.

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Heidelberg, May 2016.

[LLX07] Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 253–269. Springer, Heidelberg, June 2007.

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2009.

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, Heidelberg, February / March 2013.

[Mea86] Catherine A. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 7-9, 1986*, pages 134–137. IEEE Computer Society, 1986.

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, August 1990.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society Press, May 2013.

[MGS11] Carlos Aguilar Melchor, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, pages 648–652. IEEE, 2011.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, August 2013.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, Heidelberg, February 2005.

[Nyb96] Kaisa Nyberg. Fast accumulated hashing. In Dieter Gollmann, editor, *FSE'96*, volume 1039 of *LNCS*, pages 83–87. Springer, Heidelberg, February 1996.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.

[Pei16] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016.

[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, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Heidelberg, August 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020.

[PS14] Henrich Christopher Pöhls and Kai Samelin. On updatable redactable signatures. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14*, volume 8479 of *LNCS*, pages 457–475. Springer, Heidelberg, June 2014.

[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*, pages 515–530. USENIX Association, August 2015.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[SS79] Richard Schroeppel and Adi Shamir. A T sˆ2 = oˆn) time/space tradeoff for certain np-complete problems. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 328–336. IEEE Computer Society, 1979.

[Ste94a] Jacques Stern. Designing identification schemes with keys of short size. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 164–173. Springer, Heidelberg, August 1994.

[Ste94b] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, Heidelberg, August 1994.

[Vér97] Pascal Véron. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 8(1):57–69, 1997.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002.

[WWP08] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. Improvement of a dynamic accumulator at ICICS 07 and its application in multi-user keyword-based retrieval on encrypted data. In *Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, APSCC 2008, Yilan, Taiwan, 9-12 December 2008*, pages 1381–1386. IEEE Computer Society, 2008.

[YAY+18] Zuoxia Yu, Man Ho Au, Rupeng Yang, Junzuo Lai, and Qiuliang Xu. Lattice-based universal accumulator with nonmembership arguments. In Willy Susilo and Guomin Yang, editors, *ACISP 18*, volume 10946 of *LNCS*, pages 502–519. Springer, Heidelberg, July 2018.