TECHNISCHE
UNIVERSITÄT
DARMSTADT

Darmstadt University of Technology
Department of Computer Science
Cryptography and Computeralgebra

Master Thesis
May 2015

Post-Quantum Commitment Schemes

Nabil Alkeilani Alkadri
Darmstadt University of Technology
Department of Mathematics

Supervised by Prof. Dr. Johannes Buchmann
Dr. Denise Demirel
Rachid El Bansarkhani

Contents

Table of Contents 4

Introduction 5

1 Preliminaries 9

1.1 Commitment Schemes 10

1.1.1 Applications 13

1.2 Lattices and Cryptography 15

1.2.1 Basic Definitions 15

1.2.2 Lattice Problems 19

1.3 Cryptographic Primitives 22

2 Commitment Schemes from General Constructions 27

2.1 Commitments from One-Way Functions 29

2.1.1 Statistically Binding from [Nao91] 29

2.1.2 Statistically Hiding from [HNO+09] 32

2.2 Commitments from One-Way Permutations 38

2.2.1 Perfectly Binding from [GK96] 38

2.2.2 Perfectly Hiding from [NOVY98] 39

2.2.3 Perfectly Hiding from [TPY08] 41
2.3 Commitments from Collision-Free Hashing 44
2.3.1 Statistically Hiding from [HM96] 44
2.3.2 Statistically Hiding from [DPP97] 46
2.3.3 Statistically Hiding from [DPP98] 49
2.3.4 Sufficient Conditions for Collision-Free Hashing 50
2.4 Commitments from Public-Key Cryptosystems 54
2.4.1 Statistically Binding from [CD04] 54
2.5 Commitments from Different Protocols 56
2.5.1 Perfectly Hiding from Σ-Protocols 56
2.5.2 Statistically Hiding from PIR Protocols 58

3 Post-Quantum Commitment Schemes **62**

3.1 Statistically Hiding Commitments from (Ring-) SIS 63
3.1.1 (Ideal-) Lattice-Based Hash Functions 64
3.1.2 Commitments from (Ring-) SIS 66
3.2 Statistically Binding Commitments from LWE-Based Encryption . . 69
3.3 Perfectly Binding Commitments from LPN 71
3.4 Perfectly Binding Commitments from RLWE 72

4 Comparing Commitment Schemes **74**

4.1 Comparison Between General Constructions 75
4.1.1 Unconditionally Hiding 76
4.1.2 Unconditionally Binding 79

4.2 Comparison Between Concrete Constructions 80
4.2.1 Unconditionally Hiding 80
4.2.2 Unconditionally Binding 82
4.3 Comparing All Together 85
4.3.1 Unconditionally Hiding 85
4.3.2 Unconditionally Binding 88

Conclusion and Future Work **89**

References **97**

Index **99**

Introduction

Commitment schemes are fundamental building blocks in constructions of many cryptographic protocols. An intuitive description of a commitment can be illustrated using a lockable box. In a first phase, some (secret) information is committed to by putting it into a box, locking the box using its key, and giving it away. As long as the box is locked, the information inside the box cannot be guessed nor it can be modified. Later, the content of the box is revealed by opening the box using the key, and extracting the information inside the box. Therefore, a commitment scheme allows committing to some value, while keeping it secret and unmodified. Following the abstract viewpoint of a lockable box, commitments hold two fundamental properties called hiding and binding. The hiding property indicates that the information inside the locked box cannot be guessed, whereas the binding property indicates that the information cannot be modified. In addition to hiding and binding, commitment schemes can provide more properties, such as universal composability, non-malleability, or trapdoor property.

Commitments were introduced by Blum in [Blu82], who implicitly used them in order to flip a coin by telephone. Commitments were also used implicitly in the work of [SRA81] on mental poker, in order to generate a fair deal of cards between two dishonest players without using any cards. Commitments have many other applications and are used in various cryptographic protocols.

Since their introduction, many commitment schemes were suggested based on number theoretic problems. One example is the commitment scheme introduced in [Blu82], which is based on the hardness of factoring large integers. Another example is the scheme introduced in [Ped92], which is based on the hardness of extracting discrete logarithms. Commitment schemes were also suggested using more generic complexity assumptions. For example, the commitment scheme introduced in [Nao91], can be implemented using any pseudorandom generator. An additional example is the scheme introduced in [DPP98], which employs any family of collision resistant hash functions. Furthermore, there are commitment schemes

based on other computationally hard problems such as the schemes introduced in [KTX08, XXW13], which are based on lattice problems, or the scheme introduced in [JKPT12], which is based on the hardness of decoding random linear codes.

However, commitment schemes based on number theoretic problems will become insecure as soon as large enough quantum computers are built. This is due to Shor's algorithm [Sho97], which can be used to solve number theoretic problems such as integer factorization and discrete logarithms in polynomial time on quantum computers. Despite this fact, important classes of cryptography such as code-based and lattice-based cryptography are believed to provide security even under quantum attacks.

About This Thesis

In the first part of this thesis, we give a survey on post-quantum commitment schemes, i.e., commitment schemes that run on conventional computers, and whose security is believed to hold up against quantum computers. The survey includes general constructions of commitment schemes that use generic cryptographic primitives or other cryptographic protocols. Furthermore, the survey includes concrete commitment schemes based on computational problems that are believed to remain hard even under quantum attacks. As mentioned above, commitment schemes can have more properties than hiding and binding. However, in this thesis we concentrate only on these two fundamental properties, since they are sufficient for many applications including coin flipping and mental poker (see Subsection 1.1.1).

In the second part of this thesis, we compare the commitment schemes included in the survey. The purpose of the comparison is to obtain commitment schemes that are practical and efficient. More precisely, we first compare the general constructions of commitment schemes, and the concrete commitment schemes among each other. Afterwards, we compare the resulting candidates with each other to determine the most efficient commitment schemes.

Organization

Chapter 1 gives a formal definition of commitment schemes and their fundamental properties, and provides some applications. Additionally, Chapter 1 covers the computational problems and cryptographic primitives required for the survey. The first part of this thesis is covered by Chapter 2 and 3. Chapter 2 gives the survey on

general constructions of commitment schemes, while Chapter 3 includes the survey on concrete commitment schemes. Chapter 4 covers the second part of this thesis, which provides the comparison of commitment schemes given in Chapter 2 and 3. Finally, a conclusion of this thesis is given in Chapter 5.

Acknowledgments

I would like to thank my supervisors Dr. Denise Demirel and Rachid El Bansarkhani for creative comments and remarks to preliminary drafts of this thesis.

During preparation of this thesis, I made efforts and attempts in order to come up with new results related to the topic of this thesis. For this purpose, I had discussions with Dr. Özgür Dagdelen and Florian Göpfert, in addition to my supervisors Dr. Denise Demirel and Rachid El Bansarkhani. I would like to thank them all.

Finally, I am grateful to Professor Johannes Buchmann for motivating me to bring new results to the field of post-quantum cryptography.

Declaration

I hereby declare that the content of this thesis is the result of my own work and effort. Where other contributions including published or unpublished material have been used, they have been acknowledged and referenced.

Nabil Alkeilani Alkadri May 2015

Chapter 1

Preliminaries

This chapter covers the necessary mathematical and cryptographic background required throughout this work. We start by establishing some general notations. Then we review the definition of commitment schemes with their fundamental security properties, and view some applications. After that, we review the definition of general and ideal lattices followed by lattice computational problems. Finally, we recall the required cryptographic primitives.

Notations

We let $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ denote the set of natural numbers (with zero), integers, rationals, reals, and complex numbers, respectively. For a positive integer k, we let $[k]$ denote the set $\{1, 2, \ldots, k\}$. We denote vectors with bold lower-case letters (e.g., $\mathbf{a}$), which are assumed to be in column form. Furthermore, we denote matrices with bold upper-case letters (e.g., $\mathbf{A}$), and we write $D^{n \times m}$ for the set of all $n \times m$ matrices over a domain D. For two vectors $\mathbf{x}, \mathbf{y}$ over some finite domain, we denote their concatenation by $\mathbf{x} \| \mathbf{y}$. For two bit strings $\mathbf{x}, \mathbf{y}$, we write $\mathbf{x} \oplus \mathbf{y}$ to denote the bitwise exclusive-OR (XOR) operation on $\mathbf{x}, \mathbf{y}$.

For a positive integer n, we write $\mathbb{Z}_n$ to denote the set $\mathbb{Z}/n\mathbb{Z}$. We write $\lfloor \cdot \rceil$ for rounding to the nearest integer. Furthermore, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ means rounding up to the next integer and rounding down, respectively. All logarithms appear in this work are to base 2, i.e., $\log x = \log_2 x$, for all $x > 0$.

For a vector $\mathbf{x}$ with entries x_i over some domain, the ℓ_p norm is defined as $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{1/p}$ for $1 \leq p < \infty$, and the ℓ_∞ norm is defined as $\|\mathbf{x}\|_\infty = \max_i |x_i|$. For two vectors $\mathbf{x}, \mathbf{y}$ over some domain, we let $\langle \mathbf{x}, \mathbf{y} \rangle$ denote the inner product of the vectors $\mathbf{x}, \mathbf{y}$.

The term *negligible* describes any function $f : \mathbb{N} \longrightarrow \mathbb{R}$ that decreases faster than the reciprocal of any polynomial p, i.e., if there exists an $n_0 \in \mathbb{N}$ such that for all $n > n_0$, it holds $f(n) < \frac{1}{p(n)}$. With $negl(n)$, we denote a negligible function $f(n)$. A probability is called *overwhelming* if it is $1 - \varepsilon$, where ε is some negligible function.

The *statistical distance* between two distributions X, Y over a countable domain D is defined by the value $\Delta(X,Y) = \frac{1}{2}\sum_{d \in D} |\text{prob}[X = d] - \text{prob}[Y = d]|$. Two distributions X, Y indexed by a positive integer n are called *statistically close* if their statistical distance is negligible in n.

For some distribution $\mathcal{D}$ over some finite set S, we write $s \leftarrow \mathcal{D}$ to choose s from S according to the distribution $\mathcal{D}$, and we write $s \xleftarrow{\$} S$ to choose s uniformly at random from the set S.

The growth of functions $f(n), g(n)$ is described with the standard asymptotic notation: The notation $f(n) \in O(g(n))$ means that the magnitude of $f(n)$ is upper-bounded by a positive constant times $g(n)$, for all large n. The notation $f(n) \in o(g(n))$ means that the magnitude of $f(n)$ is upper-bounded by every positive constant times $g(n)$, i.e., for all $c > 0$, there exists an n_0 such that $|f(n)| \leq c \cdot |g(n)|$, for all $n \geq n_0$. The notation $f(n) \in \Omega(g(n))$ stands for any function $f(n)$, whose magnitude is at least as large as a positive constant times $g(n)$, for all large n. The notation $f(n) \in \Theta(g(n))$ means both $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$, i.e., there exist positive constants c, c' and n_0 such that $c \cdot g(n) \leq |f(n)| \leq c' \cdot g(n)$, for all $n \geq n_0$. Finally $f(n) \in \omega(g(n))$ means $g(n) \in o(f(n))$.

For a positive integer n, we let $poly(n)$ denote a polynomially bounded function $f(n) \in O(n^c)$, for some constant $c > 1$.

1.1 Commitment Schemes

This section gives the definition of commitment schemes, provides basic classifications of them, and describes their fundamental security properties. In addition, it gives some applications of commitment schemes.

Commitment schemes are basic ingredients in the theory and practice of secure cryptographic protocols. They were introduced by Blum in [Blu82] to solve the coin flipping problem. Loosely speaking, a commitment scheme is a digital analogous of a secure box. Namely, it is a method that allows a party called the *prover* to commit to a secret by putting it into a secured box, and giving it to a party called

the *verifier*. In a later stage, the prover gives the key to the box to let the verifier open the box and learn the secret.

Thus, a commitment scheme is an efficient two-phase protocol between a prover, denoted by $\mathcal{P}$, and a verifier, denoted by $\mathcal{V}$. The first phase is called *commit* phase, and the second phase is called *decommit*, *reveal*, or *open* phase. The protocol is first initialized by generating some public parameters. Then in the commit phase, $\mathcal{P}$ commit to a value, while keeping it secret. Later in the decommit phase, $\mathcal{P}$ chooses to open the commitment to $\mathcal{V}$, which verifies that the opening corresponds to only a single value determined in the commit phase.

As long as the box is locked, $\mathcal{V}$ does not gain any knowledge of the secret until the decommit phase. Thus, the protocol requires that $\mathcal{V}$ is not capable to distinguish two commitments generated from two distinct values. This requirement has to be satisfied even if $\mathcal{V}$ tries to cheat, and it is called *hiding*. Furthermore, since $\mathcal{P}$ gave away the box, the secret cannot be changed after the commit phase, i.e., there exists at most one value that $\mathcal{V}$ can later (in the decommit phase) accept as a legal opening of the commitment. This requirement has to be satisfied even if $\mathcal{P}$ tries to cheat, and it is called *binding*.

In the following we give a formal definition of commitment schemes. We basically follow [JKPT12].

Definition 1.1 (Commitment Scheme)**.** A *commitment scheme* consists of three polynomial-time algorithms (Setup,Com,Ver) such that

- Setup(1^n): A setup algorithm Setup takes as input 1^n, for a security parameter n, and outputs some public parameter pk as a public commitment key, i.e., $pk \leftarrow$Setup(1^n).
- Com(pk, m): A commitment algorithm Com takes as input a public key pk, and a message m. It outputs a commitment c, and a reveal value d, i.e., $(c, d) \leftarrow$Com(pk, m).
- Ver(pk, m, c, d): A verification algorithm Ver takes as input a public key pk, a message m, a commitment c, and a reveal value d. It returns 1 or 0 to accept or reject, respectively, i.e., $b \leftarrow$Ver(pk, m, c, d), where $b \in \{0, 1\}$.

A commitment scheme must satisfy the following basic requirement.

- **Perfect completeness**: The verification algorithm Ver outputs 1 whenever

the inputs are computed honestly, i.e.,

$$\text{prob}\left[\text{Ver}(pk, m, c, d) = 1 \mid pk \leftarrow \text{Setup}(1^n), (c, d) \leftarrow \text{Com}(pk, m)\right] = 1 \ .$$

Commitments come in two dual flavors: *unconditionally hiding but computationally binding*, and *unconditionally binding but computationally hiding*. A computationally hiding (respectively, binding) commitment indicates that the hiding (respectively, binding) property holds computationally, i.e., the verifier (respectively, prover) is restricted to be probabilistic polynomial-time (PPT) with limited computing power in order to break the hiding (respectively, binding) property. On the other hand, an unconditionally hiding (respectively, binding) commitment indicates that the hiding (respectively, binding) property holds information-theoretically, i.e., the verifier (respectively, prover) has unbounded computing power in order to break the hiding (respectively, binding) property. The power which the prover or verifier has, describes time and space complexity.

A commitment scheme cannot be unconditionally hiding and unconditionally binding simultaneously. If a commitment scheme is unconditionally hiding, then for a commitment c of a message m, where $(c, d) \leftarrow \text{Com}(pk, m; r)$ for some randomness r, there must exist an r' such that $c = c'$, where $(c', d') \leftarrow \text{Com}(pk, m'; r')$. Therefore, with unlimited computing power, one can find this r', and open the commitment c to m' instead of m. On the other hand, if a commitment scheme is unconditionally binding, then it is almost impossible to find two distinct messages m, m' such that $c = c'$, where $(c, d) \leftarrow \text{Com}(pk, m; r)$, $(c', d') \leftarrow \text{Com}(pk, m'; r')$ for some r, r'. Hence, there exists an almost unique r such that either $(c, d) \leftarrow \text{Com}(pk, m; r)$ or $(c, d) \leftarrow \text{Com}(pk, m'; r)$. Therefore, with unlimited computing power, one can find this r, and hence the message m. Thus, at least one of both prover and verifier must be computationally bounded.

Remark 1.1. We stress that the unconditionally hiding or binding property holds *statistically* or *perfectly*. In this work, we mention explicitly if the hiding or binding property holds statistically or perfectly.

Hereafter, we give a formal definition of the binding and hiding property. We basically follow [JKPT12] with some details from [DFS04].

Definition 1.2 (Binding Property)**.** A commitment scheme (Setup,Com,Ver) is *statistically* (respectively, *computationally*) *binding* if no (respectively, PPT) forger $\mathcal{P}^*$ can come up with a commitment and two different openings with noticeable (non-obvious) probability, i.e., for every (respectively, PPT) forger $\mathcal{P}^*$, and every two

distinct messages m, m' from the message space, there exists a negligible function $\varepsilon(n)$ such that

$$\text{prob}\left[\text{Ver}(pk, m, c, d) = \text{Ver}(pk, m', c, d') \mid pk \leftarrow \text{Setup}(1^n), (c, m, d, m', d') \leftarrow \mathcal{P}^*(pk)\right] \leq \varepsilon(n),$$

where $(c, d) \leftarrow \text{Com}(pk, m)$, $(c, d') \leftarrow \text{Com}(pk, m')$, and $pk \leftarrow \text{Setup}(1^n)$.

The scheme is *perfectly binding* if the following is satisfied: With overwhelming probability over the choice of the public key $pk \leftarrow \text{Setup}(1^n)$, it holds

$$\big(\text{Ver}(pk, m, c, d) = 1 \ \wedge \ \text{Ver}(pk, m', c, d') = 1\big) \Rightarrow m = m' \ .$$

Definition 1.3 (Hiding Property)**.** A commitment scheme (Setup,Com,Ver) is *statistically* (respectively, *computationally*) *hiding* if no (respectively, PPT) distinguisher $\mathcal{V}^*$ is able to distinguish (pk, c) and (pk, c'), for two distinct messages m, m', with non-negligible advantage. This means, for every (respectively, PPT) distinguisher $\mathcal{V}^*$, and every two distinct messages m, m' from the message space, there exists a negligible function $\varepsilon(n)$ such that the statistical distance between (pk, c) and (pk, c') is $\varepsilon(n)$, i.e.,

$$\Delta\big((pk, c), (pk, c')\big) \leq \varepsilon(n),$$

where $(c, d) \leftarrow \text{Com}(pk, m)$, $(c', d') \leftarrow \text{Com}(pk, m')$, and $pk \leftarrow \text{Setup}(1^n)$.

The scheme is *perfectly hiding* if (pk, c) and (pk, c') are equally distributed.

Remark 1.2. For the rest of this work, we use the term *unconditionally hiding* to denote an unconditionally hiding but computationally binding commitment scheme. Similarly, the term *unconditionally binding* indicates an unconditionally binding but computationally hiding commitment scheme.

Commitment schemes can carry *homomorphic* properties, i.e., operations on commitments to some values are reflected on the values themselves. An additively homomorphic commitment scheme, for instance, holds the following property: From commitments to m and m' , the verifier can compute a commitment to $m+m'$, such that if the prover opens this new commitment, the message $m+m'$ will be revealed.

1.1.1 Applications

Commitments have various applications to cryptographic protocols. A simple application of commitments are *digital auctions*, which exist in many variants. We

consider a simple variant, in which items are sold to the highest bidder. In the so called bidding phase, each bidder commits to a bid and sends the commitment to the auctioneer. At the end of the bidding phase, the participants publicly reveal their bids to the auctioneer, which announces the winner. The hiding property is reflected by the fact that the actual bids should be kept secret until the bidding phase is over. On the other hand, the binding property indicates that no bidder should be able to change his bid after seeing a revealed competitor's bid.

Another application of commitments is to solve the *coin flipping problem* over telephone introduced in [Blu82]. A protocol for common coin flipping between two parties $\mathcal{A}$ and $\mathcal{B}$ is constructed as follows.

1. $\mathcal{A}$ flips a secret coin $c_{\mathcal{A}}$, and commits to it.

2. $\mathcal{B}$ flips a coin $c_{\mathcal{B}}$, and publishes it.

3. $\mathcal{A}$ reveals $c_{\mathcal{A}}$, and $c_{\mathcal{A}} \oplus c_{\mathcal{B}}$ is taken as the common coin that both $\mathcal{A}$ and $\mathcal{B}$ trust to be random.

Commitment schemes can also be used as a building block in constructions of *zero-knowledge arguments* and *zero-knowledge proofs*. A construction of a zero-knowledge argument and a zero-knowledge proof based on commitments was shown in [NOVY98, GMW91], respectively. Zero-knowledge arguments and zero-knowledge proofs were introduced in [BCC88, GMR85], respectively. Roughly speaking, proving some fact in zero-knowledge is a method, which allows a prover to convince a verifier that a certain fact is true, while not revealing any additional information. In zero-knowledge arguments, the verifier can have unbounded computing power, i.e., the security is information-theoretic. However, the prover is computationally bounded, i.e., it can cheat in its proof and let the verifier accept a false statement if some cryptographic assumption is broken during the execution of the protocol. Therefore, zero-knowledge arguments require unconditionally hiding commitments. In zero-knowledge proofs, the prover can have unbounded computing power. However, the proof itself is only computationally secure. Therefore, zero-knowledge proofs require unconditionally binding commitments. For a formal definition of zero-knowledge arguments and zero-knowledge proofs, we refer to [NOVY98, Gol01].

Many other applications of commitments exist including the work of [FOO92] on *digital election schemes*, the work of [SRA81] on *mental poker*, and the work of [Eve83] on *digital contract signing*, in which two mutually suspicious parties wish to digitally exchange signatures on a contract.

Remark 1.3. We stress that all applications described above require either unconditionally hiding or unconditionally binding commitment schemes. These schemes are not required to hold additional properties.

1.2 Lattices and Cryptography

Lattices are a major source of hard problems that are believed to be secure even under quantum attacks. Almost all post-quantum commitment schemes, which we review in Chapter 3 are schemes whose security is based on lattice problems. Therefore, we allocate this section to review the definition of lattices and the required lattice computational problems. This section starts by introducing the mathematical theory of general and ideal lattices, and then it defines the required computational problems arising from lattices. We basically follow [Mic11, MR09, Xag10], with some details from [GPV08, HPS08, LM06, Pie12, LPR13].

1.2.1 Basic Definitions

A lattice is a discrete additive subgroup of the n-dimensional Euclidean space $\mathbb{R}^n$. We can view a lattice as an orderly arrangement of points in $\mathbb{R}^n$, where we put a point at the tip of each vector.

Definition 1.4 (Lattice)**.** Let $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k \in \mathbb{R}^n$ be a set of linearly independent vectors. The *lattice* $\mathcal{L}$ generated by the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ is the set of all linear combinations of $\mathbf{b}_1, \ldots, \mathbf{b}_k$ with integer coefficients, i.e.,

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_k) = \left\{ \sum_{i=1}^{k} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\} \subset \mathbb{R}^n \ .$$

The vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ are called a *basis* for the lattice $\mathcal{L}$, and they can be given by a matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k] \in \mathbb{R}^{n \times k}$. Thus, we can write

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^k\} \subset \mathbb{R}^n \ .$$

Any set of k independent vectors from $\mathbb{R}^n$ that generates the lattice $\mathcal{L}$ is a basis for $\mathcal{L}$. Moreover, if $\mathbf{U}$ is a unimodular matrix, i.e., an integer square matrix with determinant ± 1, then the bases $\mathbf{B}, \mathbf{BU}$ generate the same lattice. The *dimension* of $\mathcal{L}$ is the number of vectors in a basis for $\mathcal{L}$. A lattice is called *full dimensional* or *full rank* if $n = k$, i.e., if the lattice is n-dimensional in the Euclidean space $\mathbb{R}^n$. An *integer* lattice is a lattice whose vectors have integer coordinates.

Definition 1.5 (Dual Lattice)**.** Let $\mathcal{L}$ be a k-dimensional lattice in $\mathbb{R}^n$. The *dual* of $\mathcal{L}$, denoted by $\mathcal{L}^\vee$, is the lattice given by the set of all vectors $\mathbf{w} \in \mathbb{R}^n$ satisfying $\langle \mathbf{v}, \mathbf{w} \rangle \in \mathbb{Z}$ for all vectors $\mathbf{v} \in \mathcal{L}$, i.e.,

$$\mathcal{L}^\vee = \left\{ \mathbf{w} \in \mathbb{R}^n \mid \langle \mathbf{v}, \mathbf{w} \rangle \in \mathbb{Z}, \forall \mathbf{v} \in \mathcal{L} \right\} .$$

For any matrix $\mathbf{B} \in \mathbb{R}^{n \times k}$, it holds $\mathcal{L}(\mathbf{B})^\vee = \mathcal{L}((\mathbf{B}^{-1})^T)$.

Definition 1.6 (Fundamental Parallelepiped)**.** Let $\mathcal{L}$ be a k-dimensional lattice in $\mathbb{R}^n$, and $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ a basis for $\mathcal{L}$. The *fundamental parallelepiped* for $\mathcal{L}$ corresponding to the basis $\mathbf{B}$ is the set

$$\mathcal{F}(\mathbf{B}) = \left\{ \alpha_1 \mathbf{b}_1 + \ldots + \alpha_k \mathbf{b}_k \mid 0 \leq \alpha_i < 1, \ \forall i \in [k] \right\} = \mathbf{B}[0,1)^k .$$

Example 1.1. We illustrate in Figure 1.1 a full rank lattice in $\mathbb{R}^2$. The gray area displays a fundamental parallelepiped in dimension 2, which is spanned by the basis vectors $\mathbf{b}_1, \mathbf{b}_2$.

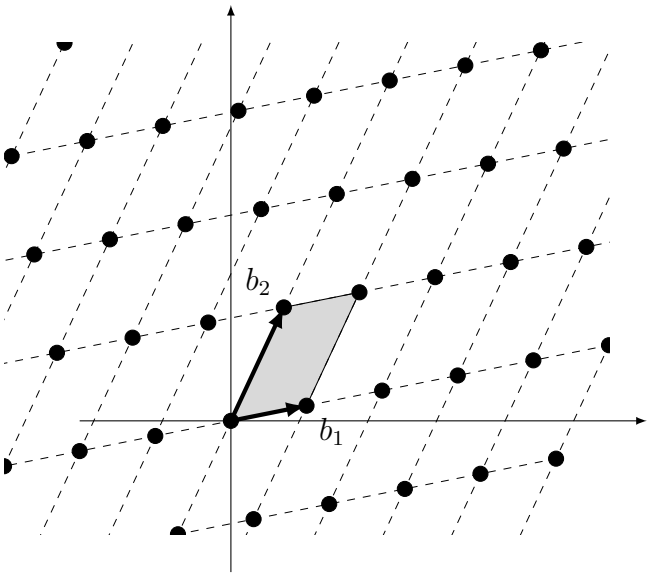

Figure 1.1: A full rank lattice in $\mathbb{R}^2$, and its fundamental parallelepiped.

Definition 1.7 (Determinant)**.** Let $\mathcal{L}$ be a k-dimensional lattice in $\mathbb{R}^n$, and $\mathbf{B}$ a basis for $\mathcal{L}$. Furthermore, let $\mathcal{F}$ be a fundamental parallelepiped for $\mathcal{L}$ corresponding to $\mathbf{B}$. The *determinant* of $\mathcal{L}$, denoted by $\det(\mathcal{L})$, is the k-dimensional volume $vol(\mathcal{F})$ of $\mathcal{F}$, i.e., $\det(\mathcal{L}) = vol(\mathcal{F}) = \sqrt{\det(\mathbf{B}^T \cdot \mathbf{B})}$.

Definition 1.8 (Successive Minima)**.** Let $\mathcal{L}$ be a k-dimensional lattice in $\mathbb{R}^n$. For all $i \in [k]$, the i-th *successive minimum* of $\mathcal{L}$, measured in some norm ℓ_p ($p \geq 1$), and denoted by $\lambda_i^p(\mathcal{L})$, is the radius $r > 0$ of the smallest ball $\mathcal{B} = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_p \leq r \right\}$ in $\mathbb{R}^n$ centered at the origin, which contains i linearly independent vectors of $\mathcal{L}$.

As a special case, the first minimum $\lambda_1^p(\mathcal{L})$ is the length of the shortest non-zero lattice vector, i.e.,

$$\lambda_1^p(\mathcal{L}) = \min_{\mathbf{0}\neq\mathbf{v}\in\mathcal{L}} \|\mathbf{v}\|_p \ .$$

The shortest non-zero lattice vector $\lambda_1^p(\mathcal{L})$ is equal to the *minimum distance* $\lambda^p(\mathcal{L})$ between any two distinct lattice points, i.e.,

$$\lambda_1^p(\mathcal{L}) = \lambda^p(\mathcal{L}) = \min\left\{\|\mathbf{x}-\mathbf{y}\|_p \mid \mathbf{x},\mathbf{y}\in\mathcal{L}, \mathbf{x}\neq\mathbf{y}\right\} \ .$$

Example 1.2. We consider the integer lattice $\mathcal{L} = \mathbb{Z}^3$ with basis vectors $\mathbf{b}_1 = (2,0,0), \mathbf{b}_2 = (0,2,0)$, and $\mathbf{b}_3 = (1,1,1)$. We determine the successive minima of $\mathbb{Z}^3$ measured in ℓ_∞ and ℓ_1 norm.

- In ℓ_∞ norm: Since the basis vectors $\mathbf{b}_1, \mathbf{b}_2$, and $\mathbf{b}_3$ are integer vectors, the smallest possible ball has radius $r = 1$. We can achieve norm 1 by the vectors $\mathbf{b}_3, \mathbf{b}_1 - \mathbf{b}_3$, and $\mathbf{b}_2 - \mathbf{b}_3$, which are linearly independent. Thus, the successive minima of $\mathbb{Z}^3$ in ℓ_∞ norm are $\lambda_1^\infty(\mathbb{Z}^3) = \lambda_2^\infty(\mathbb{Z}^3) = \lambda_3^\infty(\mathbb{Z}^3) = 1$.

- In ℓ_1 norm: The basis vectors are integer vectors so that the smallest possible value in ℓ_1 norm is 1. We can easily verify that non of the vectors with norm 1 is in the given lattice. The next smallest possible value in ℓ_1 norm is 2. We can achieve norm 2 by the vectors $\mathbf{b}_1, \mathbf{b}_2$, and $\mathbf{b}_1 + \mathbf{b}_2 - 2\mathbf{b}_3$, which are linearly independent. Thus, the successive minima of $\mathbb{Z}^3$ in ℓ_1 norm are $\lambda_1^1(\mathbb{Z}^3) = \lambda_2^1(\mathbb{Z}^3) = \lambda_3^1(\mathbb{Z}^3) = 2$.

Random lattices

A *q-ary lattice* is a lattice $\mathcal{L}$ that satisfies $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$, for some (prime) integer q. The most common two m-dimensional q-ary lattices used in lattice cryptography are denoted by $\Lambda_q(\mathbf{A}), \Lambda_q^\vee(\mathbf{A})$, for a given matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, where n is considered as the main security parameter, m is typically a multiple of n, and q is prime. The lattices $\Lambda_q(\mathbf{A}), \Lambda_q^\vee(\mathbf{A})$ are defined as follows.

$$\Lambda_q(\mathbf{A}) = \left\{\mathbf{x}\in\mathbb{Z}^m \ : \ \mathbf{x} = \mathbf{A}^T\mathbf{s} \pmod q, \text{ for some } \mathbf{s}\in\mathbb{Z}^n\right\}, \tag{1.1}$$

$$\Lambda_q^\vee(\mathbf{A}) = \left\{\mathbf{x}\in\mathbb{Z}^m \ : \ \mathbf{A}\mathbf{x} = \mathbf{0} \pmod q\right\} \ . \tag{1.2}$$

The lattice $\Lambda_q(\mathbf{A})$ is generated by the rows of the matrix $\mathbf{A}$ modulo q, while the lattice $\Lambda_q^\vee(\mathbf{A})$ contains all vectors that are orthogonal modulo q to the rows of $\mathbf{A}$. Both lattices are dual to each other, up to a scaling factor q. A random lattice is obtained by picking the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ uniformly at random.

Gaussian measures

For any $s > 0$, the *Gaussian function* on $\mathbb{R}^n$ centered at $\mathbf{c} \in \mathbb{R}^n$ is defined as follows.

$$\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x} - \mathbf{c}\|_p^2/s^2), \text{ for all } \mathbf{x} \in \mathbb{R}^n \ .$$

The *standard deviation* is given by $\sigma = s/\sqrt{2\pi}$.

The *discrete Gaussian distribution* over an n-dimensional lattice $\mathcal{L}$ is defined as follows.

$$D_{\mathcal{L},s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\mathcal{L})}, \text{ for all } \mathbf{x} \in \mathcal{L} \ .$$

The subscript $\mathbf{c}$ is taken to be $\mathbf{0}$ when omitted.

The *normal distribution* with mean $\mathbf{0}$ and standard deviation σ is the distribution on $\mathbb{R}^n$ having density function

$$\frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot \exp(-\|\mathbf{x}\|_p^2/2\sigma^2), \text{ for all } \mathbf{x} \in \mathbb{R}^n \ .$$

Ideal lattices

Ideal lattices are lattices with a certain algebraic structure that leads to efficient and practical cryptographic constructions with compact description such as collision resistant hash functions.

We recall that an *ideal* of a ring A (with unity) is an additive subgroup that is closed under multiplication by elements of A. For a ring element $a \in A$, we define (a) to be the set of all multiples of a, i.e., $(a) = \{az \ : \ z \in A\}$. We note that (a) is an ideal of A, and it is called *principal ideal*. We consider the polynomial ring $\mathbb{Z}[x]$ in a variable x, and define $R = \mathbb{Z}[x]/(f(x))$ to be the *polynomial factor ring* for a polynomial $f(x) \in \mathbb{Z}[x]$ of degree n, i.e., the elements of R are residue classes of polynomials of degree less than n with integer coefficients. Each element $g(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} \in R$ corresponds to an n-dimensional integer vector $(a_0, a_1, \ldots, a_{n-1})^T \in \mathbb{Z}^n$ by using the bijection $R \longrightarrow \mathbb{Z}^n$. By writing g, we mean both representations simultaneously. Therefore, the ideal $(g) \subseteq R$ corresponds to a lattice in $\mathbb{Z}^n$, and it is called *ideal lattice*. If the polynomial $f(x)$ is monic and irreducible, then the ideal (g) corresponds to a full rank lattice in $\mathbb{Z}^n$.

For any polynomial $g \in \mathbb{Z}[x]$, the norm of g with respect to the modulus f is defined as $\|g\|_f = \|g \mod f\|_\infty$. However, when reducing a polynomial g modulo f, the maximum coefficient of g can increase, and hence the norm $\|g\|_f$ could be much

greater than $\|g\|_\infty$. Thus, the polynomial f must have a property that prevents such an exponential growth of coefficients. This property is called the *expansion factor* of f, and it is defined as follows.

$$EF(f,k) = \max_{g\in\mathbb{Z}[x],deg(g)\leq k(deg(f)-1)} \|g\|_f/\|g\|_\infty \ . \tag{1.3}$$

Example 1.3. We can easily verify that the expansion factor of the polynomial x^n+1 is given by the bound $EF(x^n+1,k) \leq k$.

We let m be a positive integer, and $\zeta \in \mathbb{C}$ a *primitive m-th root of unity*, i.e., $\zeta^m = 1$, and $\zeta^k \neq 1$ for all $k \in [m-1]$. The m-th *cyclotomic polynomial* is defined as

$$\Phi_m(x) = \prod_{i\in\mathbb{Z}_m^*} (x-\zeta^i) \ .$$

The values ζ^i run over all the primitive m-th roots of unity in $\mathbb{C}$. Thus, the polynomial $\Phi_m(x)$ has degree $n = \varphi(m)$, where $\varphi(\cdot)$ is Euler's totient function. In addition, the polynomial $\Phi_m(x)$ is monic and irreducible. Furthermore, it holds $\Phi_m(x) \in \mathbb{Z}[x]$. In particular, $\Phi_m(x)$ is the *minimal polynomial* of ζ, i.e., $\Phi_m(x)$ is the unique monic irreducible polynomial from $\mathbb{Q}[x]$ of minimal degree having ζ as a root.

We consider the ring $R = \mathbb{Z}[x]/(\Phi_m(x))$, and the quotient ring $R_q = R/qR$, for some prime integer q. As described above, a polynomial $g(x) = a_0+a_1x+\ldots+a_{n-1}x^{n-1} \in R_q$ can be represented as a vector $(a_0,a_1,\ldots,a_{n-1})^T \in \mathbb{Z}_q^n$. We consider the values $\hat{a}_i$ that the polynomial $g(x)$ assigns on all primitive m-th roots of unity modulo q, i.e., $\hat{a}_i = g(\zeta^i) \mod q$, for all $i \in \mathbb{Z}_m^*$. The vector $(\hat{a}_0,\hat{a}_1,\ldots,\hat{a}_{n-1})^T \in \mathbb{Z}_q^n$ is called the *unique chinese remainder representation* (or *CRT representation*) of the polynomial $g(x)$ in R_q. Addition and multiplication in R_q can be performed coordinate-wise using the CRT representation.

1.2.2 Lattice Problems

The fact that the successive minima of a lattice cannot be efficiently computed, gives raise to well known hard computational problems on lattices. The best known polynomial-time approximation algorithms to solve any lattice problem only achieve approximation factors almost exponential in the dimension n of the lattice. In the following we consider only the lattice computational problems that are related to this work.

Definition 1.9 (Shortest Vector Problem (SVP))**.** An input to SVP^p is a basis $\mathbf{B}$ of a lattice $\mathcal{L}$. The goal is to find a shortest non-zero vector in $\mathcal{L}$, measured in ℓ_p norm, i.e., to output a lattice vector $\mathbf{v} \neq \mathbf{0}$ such that $\|\mathbf{v}\|_p = \lambda^p(\mathcal{L})$.

Remark 1.4. There may be more than one shortest non-zero vector in a lattice, i.e., SVP may not have a unique solution. For example, in the integer lattice $\mathbb{Z}^2$, the vectors $(0, \pm 1)$ and $(\pm 1, 0)$ are all solutions to SVP^1, SVP^2, and SVP^∞.

Definition 1.10 (Gap Version of SVP (GapSVP))**.** For a gap function $\gamma = poly(n)$, an instance of GapSVP^p_γ is a pair $(\mathbf{B}, d)$, where $\mathbf{B}$ is a basis of a lattice $\mathcal{L}$, and $d \in \mathbb{Q}$. In YES instance there exists a non-zero vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\|_p \leq d$, i.e., $\lambda^p_1(\mathcal{L}) \leq d$. In NO instance it holds $\|\mathbf{v}\|_p > \gamma d$, for any non-zero vector $\mathbf{v} \in \mathcal{L}$, i.e., $\lambda^p_1(\mathcal{L}) > \gamma d$.

The GapSVP is the decision version of the shortest vector problem (SVP).

Definition 1.11 (Shortest Independent Vectors Problem (SIVP))**.** An input to SIVP^p_γ is a basis $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L}$. The goal is to find n linearly independent lattice vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathcal{L}$ such that $\max_{1 \leq i \leq n} \|\mathbf{v}_i\|_p \leq \gamma \cdot \lambda^p_n(\mathcal{L})$, where $\gamma = poly(n)$ is an approximation factor.

The case $\gamma = 1$ in the above definition corresponds to solving the problem exactly, i.e., finding n linearly independent lattice vectors such that $\max_{1 \leq i \leq n} \|\mathbf{v}_i\|_p$ is as small as possible.

Definition 1.12 (Small Integer Solution (SIS) Problem)**.** An input to $\text{SIS}^p_{q,m,\beta}$ is a matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}_q$, where q, m, n are fixed positive integers, and β is a positive real. The goal is to find a non-zero integer vector $\mathbf{x} \in \mathbb{Z}^m$ such that

$$\mathbf{A}\mathbf{x} \equiv \mathbf{0} \pmod{q}, \quad \text{and} \quad \|\mathbf{x}\|_p \leq \beta \ .$$

The $\text{SIS}^p_{q,m,\beta}$ problem is indeed a lattice problem for the q-ary lattice $\Lambda^\perp_q(\mathbf{A})$ defined in Equation 1.2, where given a uniform matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}_q$, the goal is to find a non-zero lattice vector $\mathbf{x} \in \Lambda^\perp_q(\mathbf{A})$ such that $\|\mathbf{x}\|_p \leq \beta$.

There are several reductions from the worst case of lattice problems to the average case of $\text{SIS}^p_{q,m,\beta}$ for some polynomially-bounded functions $m = poly(n), q = poly(n), \beta = poly(n)$, and $\gamma = poly(n)$. For instance, the work of [MR07] shows a probabilistic polynomial-time reduction from GapSVP^2_γ in the worst case to $\text{SIS}^2_{q,m,\beta}$ on the average case. The most recent work was shown in [MP13]. This work gives a probabilistic polynomial-time reduction from solving SIVP^p_γ in the worst case to solving $\text{SIS}^p_{q,m,\beta}$ on average.

Definition 1.13 (Ring Variant of SIS (Ring-SIS))**.** An input to $\text{Ring-SIS}^p_{q,m,\beta}$ is a tuple $[a_1, \ldots, a_m] \in R^m_q$, where q, m, n are fixed positive integers, $R = \mathbb{Z}[x]/(f(x))$

for a monic polynomial $f \in \mathbb{Z}[x]$ of degree n, $R_q = R/qR$, and β is a positive real. The goal is to find a non-zero vector $\mathbf{x} = [x_1, \ldots, x_m] \in R^m$ such that

$$\sum_{i=1}^{m} a_i x_i \equiv 0 \pmod{q}, \quad \text{and} \quad \|\mathbf{x}\|_p \leq \beta \ .$$

Given that the polynomial f is irreducible, the work of [LM06] gives a polynomial-time reduction from the worst case of SVP_γ^∞ on ideal lattices in R to the average case of $\text{Ring-SIS}_{q,m,\beta}^\infty$, for some bounded m, q, β, γ.

Definition 1.14 (Learning With Errors (LWE) Problem)**.** Let $n, q = poly(n) \geq 2$ be positive integers, $\mathbf{s} \in \mathbb{Z}_q^n$ a vector, and χ an error distribution over $\mathbb{Z}_q$. Define the distribution $A_{\mathbf{s},\chi}$ obtained by choosing a vector $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$, a noise $e \leftarrow \chi$, and outputting the pair $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$. The *decisional* $\text{LWE}_{q,\chi}$ problem is to distinguish (with non-negligible probability) $\ell = poly(n)$ independent samples chosen according to the distribution $A_{\mathbf{s},\chi}$ from ℓ samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. The *search* $\text{LWE}_{q,\chi}$ problem is to find the secret $\mathbf{s}$ given ℓ independent samples from $A_{\mathbf{s},\chi}$.

The decisional $\text{LWE}_{q,\chi}$ problem is indeed a lattice problem for the q-ary lattice defined in Equation 1.1, where given a uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, and a vector $\mathbf{x} \in \mathbb{Z}_q^m$, the goal is to distinguish between the case that $\mathbf{x}$ is chosen uniformly from $\mathbb{Z}_q^m$, and the case in which $\mathbf{x}$ is chosen by perturbing each coordinate of a random point in the lattice $\Lambda_q(\mathbf{A}^T)$ using the error distribution χ.

The LWE problem was introduced in [Reg09]. This work shows a reduction from the worst case of $\text{LWE}_{q,\chi}$ to the average case of $\text{LWE}_{q,\chi}$. Moreover, the work of [Reg09] gives a search to decision reduction, and shows that for certain moduli q, and Gaussian error distributions χ, $\text{LWE}_{q,\chi}$ is as hard as solving SIVP_γ^2 and GapSVP_γ^2 in the worst case, for some bounded approximation factor γ.

Definition 1.15 (Ring Variant of LWE (RLWE))**.** Let $R = \mathbb{Z}[x]/(\Phi_m(x))$ be the cyclotomic polynomial ring for m a power of 2, and $R_q = R/qR$ be the quotient ring for a prime integer $q = poly(n) \geq 2$ such that $q = 1 \mod m$. For an element $s \in R_q$, and an error distribution χ over R, define the distribution $A_{s,\chi}$ obtained by choosing an element $a \xleftarrow{\$} R_q$, an error term $e \leftarrow \chi$, and outputting the pair $(a, a \cdot s + e)$. The *decisional* $\text{RLWE}_{q,\chi}$ problem is to distinguish (with non-negligible probability) $\ell = poly(n)$ independent samples chosen according to the distribution $A_{s,\chi}$ from ℓ samples chosen according to the uniform distribution over $R_q \times R_q$, where $s \xleftarrow{\$} R_q$. The *search* $\text{RLWE}_{q,\chi}$ problem is to find the secret s given ℓ independent samples from the distribution $A_{s,\chi}$.

The RLWE problem was introduced in [LPR13]. This work shows that the decisional RLWE problem is as hard as the search problem. The work of [LPR13] also gives a polynomial-time reduction from the worst case of $\mathrm{SIVP}_{\gamma}^{2}$ on ideal lattices in R to the search $\mathrm{RLWE}_{q,\chi}$ problem, for some bounded approximation factor γ.

Definition 1.16 (Learning Parity with Noise (LPN) Problem)**.** For a positive integer n, and $0 < \tau < 1/2$. The *search* $\mathrm{LPN}_{n,\tau}$ problem asks to find a fixed uniformly random n-bit string $\mathbf{s}$ from $\ell = poly(n)$ samples of the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle \oplus e)$, where $\mathbf{a}$ is a uniformly random n-bit string, and $e \in \{0,1\}$ has Bernoulli distribution (Ber_{τ}) with parameter τ, i.e., $\mathrm{prob}[e = 1 \ ; \ e \leftarrow Ber_{\tau}] = \tau$. The *decisional* $\mathrm{LPN}_{n,\tau}$ problem asks to distinguish (with non-negligible probability) ℓ noisy inner products $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle \oplus e)$ from ℓ samples chosen according to the uniform distribution over $\{0,1\}^n \times \{0,1\}$.

The LPN problem is a fundamental problem in learning theory. It is equivalent to the problem of decoding random linear codes, which is one of the most important problems in coding theory. However, LPN problem is closely related to LWE problem. More precisely, LWE is a generalization of LPN to larger moduli.

1.3 Cryptographic Primitives

This section gives the definition of the fundamental cryptographic primitives required in this work. Namely, we recall the definition of one-way functions, one-way permutations, pseudorandom generators, collision resistant hash functions, public-key cryptosystems, and finally digital signature schemes. We basically follow [Gol01, KL07].

We start by one-way functions, which are the most basic primitive for cryptographic applications. Their existence is considered as the minimal complexity assumption, and a necessary condition to the existence of most known cryptographic primitives. Roughly speaking, a one-way function (OWF) f is a function that is easy to evaluate, but hard to invert, i.e., on input x, its output $y = f(x)$ can be computed efficiently. On the other hand, it is hard to find an inverse of y under f. A formal definition follows.

Definition 1.17 (One-Way Function (OWF))**.** A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is called *one-way* if the following two conditions are satisfied.

1. f **is easy to compute**: There exists a deterministic polynomial-time algorithm $\mathcal{E}$ such that on input $\mathbf{x}$, algorithm $\mathcal{E}$ outputs $f(\mathbf{x})$, i.e., $f(\mathbf{x}) \leftarrow \mathcal{E}(\mathbf{x})$.

2. f **is hard to invert**: For every probabilistic polynomial-time algorithm $\mathcal{A}$, it holds
$$\text{prob}\left[\mathcal{A}(1^n, f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))\right] \leq negl(n),$$
where the probability is taken over the uniformly random choice of $\mathbf{x}$ from $\{0,1\}^n$, and all possible internal coin tosses of $\mathcal{A}$.

A function f is *length-preserving* if the length of any image $f(\mathbf{x})$ is equal to the length of its preimage $\mathbf{x}$, i.e., if $|f(\mathbf{x})| = |\mathbf{x}|$ for all $\mathbf{x}$. A one-way permutation is a one-way function that is length-preserving and one-to-one. A formal definition follows.

Definition 1.18 (One-Way Permutation (OWP))**.** Let $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ be a one-way function that is length-preserving, and let f_n be the restriction of f to the domain $\{0,1\}^n$. The function f is called *one-way permutation* if for every n, the function f_n is a bijection.

Next we move to pseudorandomness, which is a computational relaxation of true randomness. Loosely speaking, a pseudorandom generator G is a deterministic algorithm that expands short random seeds into much longer bit sequences that look like a uniformly distributed string, i.e., although the output of G is not really random, it is computationally indistinguishable from truly random sequences. Thus, the security of a pseudorandom generator requires that every polynomial-time distinguisher $\mathcal{D}$ outputs the bit 1 with almost the same probability when given a truly random string, and when given a pseudorandom one. This output bit indicates the decision of $\mathcal{D}$, i.e., it outputs 1 for a pseudorandom string, and 0 for a truly random string. The advantage of $\mathcal{D}$ is the absolute value of the difference of the probabilities that the decision of $\mathcal{D}$ is correct given a pseudorandom string, and incorrect given a truly random string. A formal definition follows.

Definition 1.19 (Pseudorandom Generator (PRG))**.** Let $G : \{0,1\}^n \longrightarrow \{0,1\}^\ell$, for $n \in \mathbb{N}$, and $\ell = poly(n)$, be a deterministic polynomial-time computable function. The function G is called *pseudorandom generator* if the following two conditions hold.

1. **Expansion**: For every n, it holds that $\ell > n$. The function ℓ is called the *expansion factor* of G.

2. **Pseudorandomness**: For every polynomial-time distinguisher $\mathcal{D}$, it holds
$$\left|\text{prob}[\mathcal{D}(1^n, G(\mathbf{x})) = 1] - \text{prob}[\mathcal{D}(1^n, \mathbf{y}) = 1]\right| \leq negl(n),$$

where the probabilities are taken over the uniformly random choice of the seed $\mathbf{x}$ from $\{0,1\}^n$, the uniformly random choice of $\mathbf{y}$ from $\{0,1\}^\ell$, and the random coins used by $\mathcal{D}$.

We proceed to hash functions and collision resistance. In general, a family of hash functions simply takes strings of arbitrary length, and compresses them into shorter strings of some fixed length. Consequently, there must exist two distinct values x, x' that collide under the functions of the family, i.e., $H(x) = H(x')$, for any function H from the family. A family of hash functions consists of two polynomial-time algorithms denoted by KGen and Hash. The first algorithm determines a function from the family. The second algorithm is deterministic and computes the hash value of its input. A family of hash functions is collision resistant if it is infeasible for any probabilistic polynomial-time algorithm to find a collision in any function H from the family, i.e., to find a pair of distinct domain elements x, x', for which $H(x) = H(x')$. A formal definition follows.

Definition 1.20 (Collision Resistant Hash Functions)**.** A family of hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$ from $\{0,1\}^*$ to $\{0,1\}^n$ with security parameter $n \in \mathbb{N}$ is called *collision resistant* if for every probabilistic polynomial-time algorithm $\mathcal{A}$ it holds

$$\mathrm{prob}\left[\begin{array}{ll} k \leftarrow \mathrm{KGen}(1^n); & \mathbf{x} \neq \mathbf{x}' \wedge \\ (\mathbf{x}, \mathbf{x}') \leftarrow \mathcal{A}(k) & : \quad H_k(\mathbf{x}) = H_k(\mathbf{x}') \end{array}\right] \leq negl(n),$$

where the probability is taken over the choice of $k \leftarrow \mathrm{KGen}(1^n)$, and the internal coin tosses of the algorithm $\mathcal{A}$.

Next we recall the definition of public-key encryption. Public-key (or asymmetric) encryption allows a party $\mathcal{A}$ to generate a pair of keys (pk, sk), called the *public key* and the *private* or (*secret*) *key*, respectively. The public key pk is used by a party $\mathcal{B}$ to encrypt messages for $\mathcal{A}$. The party $\mathcal{A}$ uses the private key sk to decrypt the resulting ciphertexts, and to obtain the messages. The party $\mathcal{B}$ can also generate a key pair. This allows $\mathcal{A}$ to encrypt messages for $\mathcal{B}$ as well. Thus, both parties $\mathcal{A}, \mathcal{B}$ can communicate secretly even if all communications between them are monitored. A formal definition follows.

Definition 1.21 (Public-Key Encryption Scheme)**.** A *public-key encryption scheme* is a triple of probabilistic polynomial-time algorithms (KGen,Enc,Dec) such that

- $\underline{\mathrm{KGen}(1^n)}$: A key generation algorithm KGen takes as input 1^n, for a security parameter n. It outputs a pair of keys (pk, sk), i.e., $(pk, sk) \leftarrow \mathrm{KGen}(1^n)$, where pk refers to the *public key*, and sk to the *private* or (*secret*) *key*.

- $\underline{\mathrm{Enc}(1^n, pk, m)}$: An encryption algorithm Enc takes as input 1^n, a public key pk, and a message m from some underlying plaintext space $\mathcal{M}$. It outputs a ciphertext c, i.e., $c \leftarrow \mathrm{Enc}(1^n, pk, m)$.

- $\underline{\mathrm{Dec}(1^n, sk, c)}$: A decryption algorithm Dec takes as input 1^n, a private key sk, and a ciphertext c. It outputs a message m or a special symbol $\perp$ denoting failure, i.e., $m \leftarrow \mathrm{Dec}(1^n, sk, c)$ or $\perp \leftarrow \mathrm{Dec}(1^n, sk, c)$.

A public-key encryption scheme must satisfy the following basic requirement.

- For all $n \in \mathbb{N}$, all $(pk, sk) \leftarrow \mathrm{KGen}(1^n)$, every $m \in \mathcal{M}$, and all $c \leftarrow \mathrm{Enc}(1^n, pk, m)$, it holds
$$\mathrm{prob}[\mathrm{Dec}(1^n, sk, c) \neq m] \leq negl(n) \ .$$
In words, the decryption algorithm Dec always outputs correctly encrypted messages, except with possibly negligible probability.

We conclude with digital signatures. Digital signature schemes allow a *signer* $\mathcal{S}$ to digitally sign a message in such a way that any other party can verify that the message originated from $\mathcal{S}$, and has not been modified in any way. Thus, digital signature schemes establish the integrity, authenticity, and non-repudiability of the message to be signed. Like public-key cryptosystems, a digital signature scheme requires a key pair (pk, sk), where sk is private and used for signing messages, and pk is public and used for verifying the signed messages. A formal definition follows.

Definition 1.22 (Digital Signature Scheme)**.** A *digital signature scheme* is a triple of probabilistic polynomial-time algorithms (KGen,Sig,Ver) such that

- $\underline{\mathrm{KGen}(1^n)}$: A key generation algorithm KGen takes as input 1^n, for a security parameter n. It outputs a pair of keys (pk, sk), i.e., $(pk, sk) \leftarrow \mathrm{KGen}(1^n)$, where pk refers to the *public key*, and sk to the *private* or (*secret*) *key*.

- $\underline{\mathrm{Sig}(1^n, sk, m)}$: A signature algorithm Sig takes as input 1^n, a private key sk, and a message m from some underlying message space $\mathcal{M}$. It outputs a signature s, i.e., $s \leftarrow \mathrm{Sig}(1^n, sk, m)$.

- $\underline{\mathrm{Ver}(1^n, pk, s)}$: A deterministic verification algorithm Ver takes as input 1^n, a public key pk, and a signature s. It outputs a bit b, where $b = 1$ means *valid*, and $b = 0$ means *invalid*, i.e., $b \leftarrow \mathrm{Dec}(1^n, pk, s)$, where $b \in \{0, 1\}$.

A digital signature scheme must satisfy the following basic requirement.

- For all $n \in \mathbb{N}$, all $(pk, sk) \leftarrow \mathrm{KGen}(1^n)$, every $m \in \mathcal{M}$, and all $s \leftarrow \mathrm{Sig}(1^n, sk, m)$, it holds

$$\mathrm{prob}\big[\mathrm{Ver}\big(1^n, pk, s\big) = 1\big] = 1 \ .$$

  In words, the verification algorithm Ver always validates correctly signed messages.

Chapter 2

Commitment Schemes from General Constructions

In this chapter we provide a survey on general constructions of commitment schemes that use various cryptographic primitives and protocols. Such a general construction is a framework that employs cryptographic primitives or protocols to obtain a commitment scheme. In particular, we can obtain commitment schemes secure even against quantum computers by plugging cryptographic primitives or protocols secure under quantum attacks into the corresponding framework.

The works we consider in this survey are constructions of unconditionally hiding or unconditionally binding commitment schemes. In the introduction of this work, we pointed out that there are commitment schemes with additional properties, which we do not consider. We argued that the hiding and binding property are sufficient for many applications including those given in Subsection 1.1.1, which use either unconditionally hiding or unconditionally binding commitments.

The additional properties of commitment schemes we do not consider include for example the *trapdoor* property defined in [BCC88, DC03], which allows the prover to overcome the binding property and to open a commitment ambiguously. Another example is *non-malleability* defined in [DDN03], which prevents anyone from constructing a valid commitment to a related secret from a commitment of another person's secret. In particular, we are not concerned about commitment schemes with the *universal composability* property defined in [CF01], which guarantees that the commitment schemes remain secure even if they are composed with arbitrary protocols and polynomially many copies of the schemes are run concurrently. Moreover, we do not consider commitment schemes under the common reference string

model, where the prover and the verifier have access to a common string taken from a predetermined distribution during the commit phase.

In Figure 2.1 we give an overview of the constructions of commitment schemes considered in this survey in form of a tree. The leafs of the tree include the subsections where the constructions are reviewed followed by the respective original papers in which the constructions appeared. The parent nodes of the leafs include the respective cryptographic primitives or protocols, which the constructions employ. The cryptographic primitives are one-way function (OWF), one-way permutation (OWP), collision resistant hash function (CRHF), and public-key encryption (PKE). The cryptographic protocols are Σ-protocol and private information retrieval (PIR) protocol. The two roots represent the resulting schemes, where UHC and UBC stand for unconditionally hiding commitments and unconditionally binding commitments, respectively.

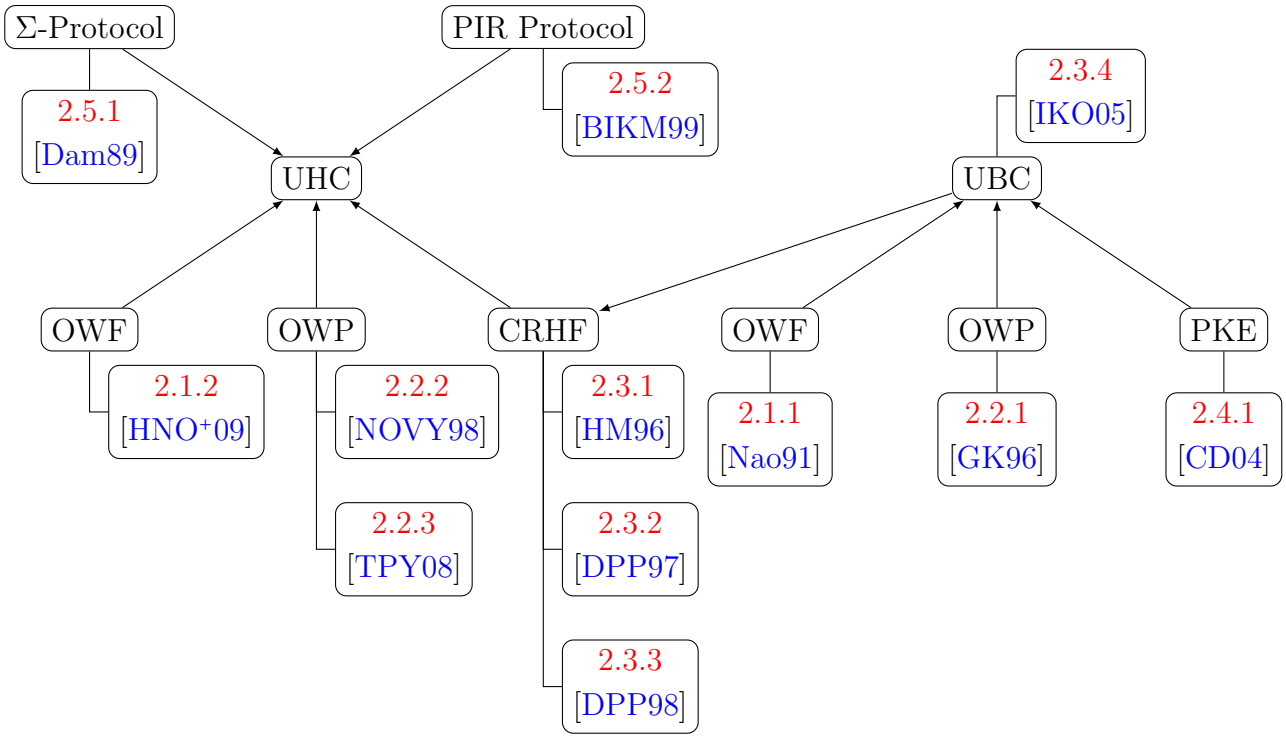

Figure 2.1: Overview of general constructions of commitment schemes.

Section 2.1 reviews the constructions of commitment schemes that use OWFs. Section 2.2 reviews the constructions that use OWPs. Section 2.3 includes those constructions that employ CRHFs, while Section 2.4 includes a construction that uses PKE. Finally, the constructions that use Σ-protocols and PIR protocols are given in Section 2.5.

2.1 Commitments from One-Way Functions

This section includes general constructions of commitment schemes proposed in [Nao91, HNO+09]. These constructions assume the existence of one-way functions.

2.1.1 Statistically Binding from [Nao91]

Next we review two constructions of statistically binding commitment schemes presented in [Nao91], one commits to single bits, and the other to bit strings. Both schemes are interactive with two rounds, and assume the existence of pseudorandom generators (PRGs), which exist if and only if one-way functions (OWFs) exist. This relation between OWFs and PRGs was shown in [HILL99]. We start by reviewing the bit commitment scheme and then the bit string commitment scheme.

We set n as the security parameter for both schemes. The security parameter guarantees the security of the used PRG for seeds of length n.

I. Bit commitment protocol

- Setup(1^n): The setup algorithm chooses any pseudorandom generator G that stretches a random seed with length n to $3n$ bits, i.e., $G\colon \{0,1\}^n \longrightarrow \{0,1\}^{3n}$.
- Com(G, b): The algorithm Com works as follows.
  Commit phase: To commit to a bit $b \in \{0,1\}$, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ act as follows.
  1. $\mathcal{V}$ selects a random bit string $\mathbf{r}$ of length $3n$ and sends it to $\mathcal{P}$.
  2. After receiving $\mathbf{r}$, $\mathcal{P}$ selects a random seed $\mathbf{x} \in \{0,1\}^n$ and returns the commitment string $\mathbf{c}$ to $\mathcal{V}$, where
  $$\mathbf{c} = \begin{cases} G(\mathbf{x}) & \text{if } b = 0 \\ G(\mathbf{x}) \oplus \mathbf{r} & \text{if } b = 1 \end{cases}.$$
  Reveal phase: To open the commitment, $\mathcal{P}$ sends b and $\mathbf{x}$ to $\mathcal{V}$.
- Ver($G, \mathbf{r}, \mathbf{c}, b, \mathbf{x}$): $\mathcal{V}$ verifies that the values $b, \mathbf{x}$ and $\mathbf{r}$ match the previously given commitment.

We provide a simple illustration of the protocol flow in Figure 2.2.

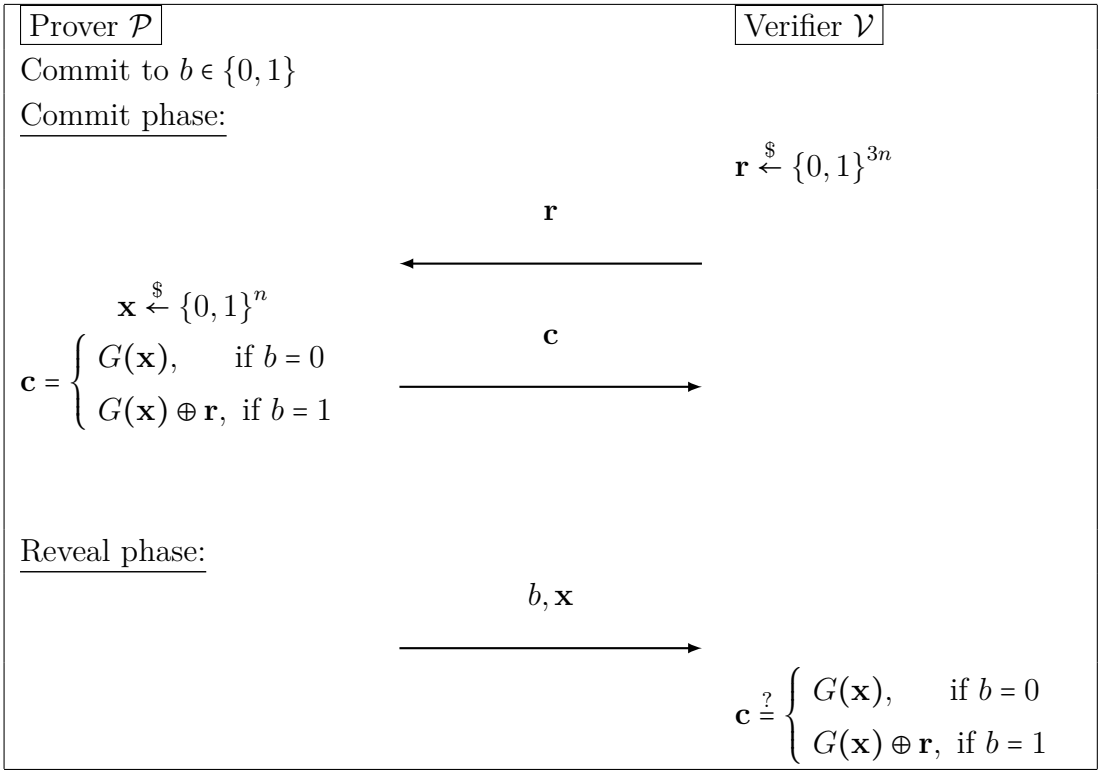

Figure 2.2: Statistically binding bit commitment protocol from [Nao91].

The computationally hiding property follows from the pseudorandomness of G, i.e., the verifier $\mathcal{V}$ cannot distinguish between both cases $b = 0$ and $b = 1$ significantly, since the output of G is pseudorandom. On the other hand, valid openings for both cases $b = 0$ and $b = 1$ with random seeds $\mathbf{x}_0$ and $\mathbf{x}_1$, respectively, require that $G(\mathbf{x}_0) = G(\mathbf{x}_1) \oplus \mathbf{r}$. Since the set $\left\{G(\mathbf{x}_0) \oplus G(\mathbf{x}_1) \mid \mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^n\right\}$ has at most 2^{2n} elements, the probability that a random $\mathbf{r} \in \{0,1\}^{3n}$ hits this set is at most $\frac{2^{2n}}{2^{3n}} = 2^{-n}$. It follows the statistically binding property.

II. Commitment protocol for bit strings

In this protocol, the prover $\mathcal{P}$ commits to an ℓ-bit string $\mathbf{b}$, where ℓ is at least linear in n. The protocol utilizes some *error-correcting code* with a large distance between code words. Essentially, error-correcting codes enable reliable delivery of messages over a noisy communication channel. This is accomplished by mapping messages to longer strings, which contain redundant elements that enable reconstruction of some corrupted bits. However, using an error-correcting code in the bit string commitment scheme proposed in [Nao91] prevents $\mathcal{P}$ from altering any bit from $\mathbf{b}$.

According to [Nao91], a constructive example of error-correcting codes is given in [Jus72].

The protocol sets $C \subset \{0,1\}^q$ to be a code of 2^ℓ words such that the Hamming distance between any two words is at least εq, where $\varepsilon > 0$, $q\log(2/(2-\varepsilon)) \geq 3n$, $2q \in O(n)$, and q/ℓ is a fixed constant. Furthermore, the protocol sets $E : \{0,1\}^\ell \longrightarrow \{0,1\}^q$ to be any efficiently computable function for mapping words in $\{0,1\}^\ell$ to C.

We write $B_i(\mathbf{x})$ to denote the i-th bit of the pseudorandom output $G(\mathbf{x})$ on seed $\mathbf{x}$. Moreover, for a k-bit string $\mathbf{r}$, where exactly q bits from $\mathbf{r}$ are 1, we let $G_{\mathbf{r}}(\mathbf{x})$ denote the string $(y_1, \ldots, y_q)$, where $y_i = B_{j(i)}(\mathbf{x})$, and $j(i)$ is the index of the i-th 1 in $\mathbf{r}$.

- Setup(1^n): The algorithm chooses any PRG, $G : \{0,1\}^n \longrightarrow \{0,1\}^{3n}$, and any efficiently computable function $E : \{0,1\}^\ell \longrightarrow \{0,1\}^q$.
- Com($G, E, \mathbf{b}$): The algorithm Com works as follows.
  Commit phase: To commit to a bit string $\mathbf{b} \in \{0,1\}^\ell$, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ act as follows.
  1. $\mathcal{V}$ selects a random bit string $\mathbf{r} \in \{0,1\}^{2q}$, where exactly q bits from $\mathbf{r}$ are 1, and sends it to $\mathcal{P}$.
  2. After receiving $\mathbf{r}$, $\mathcal{P}$ computes $E(\mathbf{b})$, selects a random seed $\mathbf{x} \in \{0,1\}^n$, computes $\mathbf{z} = E(\mathbf{b}) \oplus G_{\mathbf{r}}(\mathbf{x})$, and returns the commitment $\mathbf{c} \in \{0,1\}^{2q}$ to $\mathcal{V}$, where $\mathbf{c}$ is the string $\mathbf{z}$ along with the bits $B_i(\mathbf{x})$, for each $1 \leq i \leq 2q$ such that $r_i = 0$.

  Reveal phase: To open the commitment, $\mathcal{P}$ sends $\mathbf{b}$ and $\mathbf{x}$ to $\mathcal{V}$.
- Ver($G, E, \mathbf{r}, \mathbf{c}, \mathbf{b}, \mathbf{x}$): $\mathcal{V}$ verifies that for all $1 \leq i \leq 2q$ such that $r_i = 0$, $\mathcal{P}$ had sent the correct $B_i(\mathbf{x})$, and verifies that $E(\mathbf{b}) \oplus G_{\mathbf{r}}(\mathbf{x}) = \mathbf{z}$.

We illustrate the protocol flow in Figure 2.3.

The following theorem shown in [Nao91] establishes the security of the protocol.

Theorem 2.1. *Let G be a pseudorandom generator with a (large enough) security parameter n. Then the above bit string commitment protocol satisfies the following.*

- *For any two bit strings $\mathbf{b}, \mathbf{b}'$ the verifier $\mathcal{V}$ selects, it cannot decide (in polynomial-time) with non-negligible probability to which bit string the prover $\mathcal{P}$ has committed. Thus, the protocol is computationally hiding.*

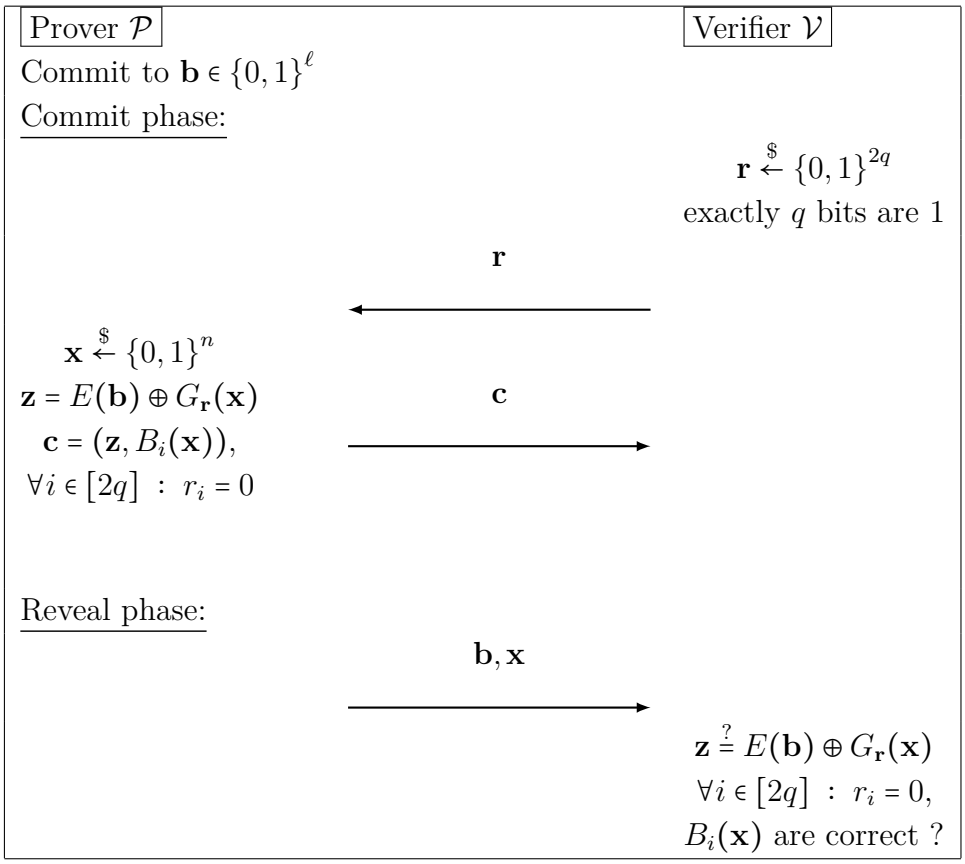

Figure 2.3: Statistically binding bit string commitment protocol from [Nao91].

- *$\mathcal{P}$ can open only one possible bit string, except with probability less than 2^{-n}. Thus, the protocol is statistically binding.*
- *For $\ell > n$, the communication cost is $O(\ell)$.*

2.1.2 Statistically Hiding from [HNO+09]

We review below the construction introduced in [HNO+09], which provides an interactive statistically hiding bit commitment scheme. The scheme employs a variant of commitment schemes called *two-phase commitment schemes* introduced in [NV06], and a family of functions called *universal one-way hash functions* introduced in [NY89]. The two-phase commitment scheme is constructed using any OWF. This commitment scheme is then used together with a family of universal one-way hash functions to construct the desired statistically hiding commitment scheme. The existence of universal one-way hash functions is also implied by the existence of OWFs as shown in [Rom90, KK05]. Hence, the entire construction is based on the existence of OWFs.

First we recall the definition of two-phase commitment schemes. Roughly speaking, a two-phase commitment scheme is an alternate variant of commitment schemes with two sequential and related stages in which the prover and the verifier interact. The transcript of the first stage is used as an input for the commitment of the second stage. In each stage, the prover commits to and reveals a value. Following is a formal definition adopted from [HNO+09].

Definition 2.1 (Two-Phase Commitment Scheme)**.** Let n be some security parameter and $(k_1 = poly(n), k_2 = poly(n))$ the message lengths. A *two-phase commitment scheme*, denoted by (P, V), consists of four interactive protocols computable in polynomial-time: the first commit stage (P_c^1, V_c^1), the first reveal stage (P_r^1, V_r^1), the second commit stage (P_c^2, V_c^2), and the second reveal stage (P_r^2, V_r^2), such that

1. In the first commit stage, P_c^1 takes as private input a bit string $\boldsymbol{\sigma}^{(1)} \in \{0,1\}^{k_1}$ and coin tosses r_P. At the end of the interaction, both P_c^1 and V_c^1 output a commitment $\mathbf{c}^{(1)}$. The commitment $\mathbf{c}^{(1)}$ is assumed to be the transcript of the first commit stage.

2. In the first reveal stage, both P_r^1 and V_r^1 take as common input the commitment $\mathbf{c}^{(1)}$, and P_r^1 takes as private input its previous coin tosses r_P. P_r^1 sends V_r^1 a pair $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\gamma}^{(1)})$, where $\boldsymbol{\gamma}^{(1)}$ is interpreted as a decommitment for $\boldsymbol{\sigma}^{(1)}$. V_r^1 accepts or rejects according to $\mathbf{c}^{(1)}, \boldsymbol{\sigma}^{(1)}$, and $\boldsymbol{\gamma}^{(1)}$. After that, both P_r^1 and V_r^1 output a string $\boldsymbol{\tau}$. The string $\boldsymbol{\tau}$ is assumed to be the transcript of the first commit stage and first reveal stage, and includes V_r^1's decision to accept or reject.

3. In the second commit stage, both P_c^2 and V_c^2 take as common input the string $\boldsymbol{\tau}$, and P_c^2 takes a private input $\boldsymbol{\sigma}^{(2)} \in \{0,1\}^{k_2}$ and its previous coin tosses r_P. At the end of the interaction, both P_c^2 and V_c^2 output a commitment $\mathbf{c}^{(2)}$. The commitment $\mathbf{c}^{(2)}$ is assumed to be the concatenation of the string $\boldsymbol{\tau}$ and the transcript of the second commit stage.

4. In the second reveal stage, both P_r^2 and V_r^2 take as common input the commitment $\mathbf{c}^{(2)}$, and P_r^2 takes as private input its previous coin tosses r_P. P_r^2 sends V_r^2 a pair $(\boldsymbol{\sigma}^{(2)}, \boldsymbol{\gamma}^{(2)})$, where $\boldsymbol{\gamma}^{(2)}$ is interpreted as a decommitment for $\boldsymbol{\sigma}^{(2)}$. V_r^1 accepts or rejects according to $\mathbf{c}^{(2)}$,$\boldsymbol{\sigma}^{(2)}$, and $\boldsymbol{\gamma}^{(2)}$.

- If the prover and the verifier follow their prescribed strategy, then the verifier will always accept (with probability 1), i.e., the scheme has perfect completeness.

- The scheme (P, V) is called *public coin*, if all messages sent by the verifier to the prover are independent random coins.

A two-phase commitment scheme is statistically hiding, if both commitment stages are statistically hiding, i.e., with overwhelming probability, the verifier cannot obtain information about the committed value before each of the reveal stages. Since the stages are run sequentially, the hiding property for the second commit stage is required to hold even given the verifier's transcript of the first stage. However, the binding property is only required to hold in one of both stages, i.e., if at least one of the two commit stages is binding. In the other (bad) stage, the prover can reveal a given commitment to two different messages. This bad stage is allowed to be determined dynamically by the prover. Moreover, the second stage is required to be statistically binding if the prover breaks the first stage. The binding property for two-phase commitment schemes is called *1-out-of-2 binding*.

Next we recall the definition of universal one-way hash functions. Universal one-way hash functions are a relaxation of collision resistant hash functions such that it is hard to find a collision (x, x') for a given (fixed) preimage x before the function is chosen from the family. The following formal definition is basically from [NY89].

Definition 2.2 (Universal One-Way Hash Functions)**.** A family of hash functions $\mathcal{F} = (\mathrm{KGen}(1^n), F)$ from a domain X to a range Y with security parameter $n \in \mathbb{N}$ is called *universal one-way*, if for any probabilistic polynomial-time algorithm $\mathcal{A} = (\mathcal{A}^{(1)}, \mathcal{A}^{(2)})$ it holds

$$\mathrm{prob}\left[\begin{array}{ll} k \leftarrow \mathrm{KGen}(1^n);\ (x, state) \leftarrow \mathcal{A}^{(1)}(1^n) & : \quad x \neq x' \ \wedge \\ x' \leftarrow \mathcal{A}^{(2)}(x, state, F_k) & \quad\ F_k(x) = F_k(x') \end{array}\right] \leq negl(n),$$

where the probability is taken over the choice of $k \leftarrow \mathrm{KGen}(1^n)$, and the internal coin tosses of the algorithm $\mathcal{A}$. The *state* variable, declared in [HNO+09], represents additional information, which the algorithm $\mathcal{A}$ is allowed to transfer between the selection of x and finding the collision.

An additional tool used for the construction of the statistically hiding commitment scheme proposed in [HNO+09] are pairwise independent hash functions introduced in [CW77]. Intuitively, a pairwise independent hash function has the property that every two distinct elements x_1, x_2 from the domain are mapped randomly and independently. A formal definition follows.

Definition 2.3 (Pairwise Independent Hash Functions)**.** A family of hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$ from a domain X to a range Y with security parameter $n \in \mathbb{N}$ is called *pairwise independent*, if for every two distinct domain elements $x_1, x_2 \in X$ and every $y_1, y_2 \in Y$ it holds

$$\mathrm{prob}\left[k \leftarrow \mathrm{KGen}(1^n) \ : \ H_k(x_1) = y_1 \ \wedge \ H_k(x_2) = y_2\right] = \frac{1}{|Y|^2} \ .$$

We proceed to the theorem presented in [HNO+09], which yields a collection of two-phase commitment schemes from any one-way function.

Theorem 2.2. *Let $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ be any one-way function. Then given the function f, a collection of $m = n^{O(1)}$ public coin two-phase commitment schemes $(P,V) = \left((P,V)_1, (P,V)_2, \ldots, (P,V)_m\right)$ with message lengths $(k_1, k_2) = (n, n)$ can be constructed in time polynomial in n, so that the following hold.*

- *There exists an index $i \in [m]$ such that the scheme $(P,V)_i$ is statistically hiding.*
- *For every index $i \in [m]$, the scheme $(P,V)_i$ is computationally 1-out-of-2 binding.*

Remark 2.1. As stated in [HNO+09], a possible way to obtain schemes for long messages is to apply the above theorem to the function $f'(\mathbf{x}_1, \ldots, \mathbf{x}_k) = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k))$, which has input length kn, for $k = poly(n)$. If the function f is one-way, then the function f' is one-way as well.

The following theorem from [HNO+09] states how to convert each two-phase commitment scheme of the collection obtained from Theorem 2.2 into a single statistically hiding commitment scheme. This is accomplished by using a family of universal one-way hash functions whose existence can be based on any one-way function [Rom90, KK05]. Thus, the transformation can be based on any one-way function.

Theorem 2.3. *There exists an efficient procedure that takes as input a security parameter n, a two-phase commitment scheme (P,V) with message lengths $(k_1, k_2) = (n, 1)$ obtained from Theorem 2.2 and a family of universal one-way hash functions $\mathcal{F} = (KGen(1^n), F)$ from $\{0,1\}^n$ to $\{0,1\}^m$, and outputs a public coin (standard) commitment scheme that is statistically hiding and computationally binding.*

Theorem 2.3 uses a commitment protocol provided in [HNO+09] that is statistically hiding and only $(1 - \delta(n))$-binding, for $\delta \geq 1/n^{O(1)}$, i.e., the binding property is weak. This protocol is then converted into a statistically hiding and computationally $negl(n)$-binding protocol. We omit the technical details of this transformation process and review next the protocol, instead.

- Setup(1^n): For a security parameter $n \in \mathbb{N}$, the setup algorithm initializes a two-phase commitment scheme (P,V) with message lengths

$(k_1, k_2) = (n, 1)$. This scheme is obtained from Theorem 2.2. Furthermore, the algorithm selects any family of universal one-way hash functions $\mathcal{F} = (\mathrm{KGen}(1^n), F)$ from $\{0,1\}^n$ to $\{0,1\}^m$, and any family of pairwise independent hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$ from $\{0,1\}^n$ to $\{0,1\}$.

- The algorithm Com works as follows.
  Commit phase: To commit to a bit $b \in \{0,1\}$, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ act as follows.

  1. $\mathcal{P}$ selects a uniform $\boldsymbol{\sigma} \in \{0,1\}^n$.
  2. $\mathcal{P}$ and $\mathcal{V}$ engage in $(P_c^1(\boldsymbol{\sigma}), V_c^1)(1^n)$, with $\mathcal{P}$ acting as P_c^1 and $\mathcal{V}$ acting as V_c^1.
     Let $\mathbf{c}^{(1)}$ be the common output of P_c^1, V_c^1 after the interaction.
  3. $\mathcal{V}$ picks a random universal one-way hash function F_k from the family $\mathcal{F}$, and sends it to $\mathcal{P}$.
  4. $\mathcal{P}$ sends $\mathbf{y} = F_k(\boldsymbol{\sigma})$ to $\mathcal{V}$.
  5. $\mathcal{V}$ flips a random coin, represented by $phase \leftarrow \{1,2\}$ [a] , and sends $phase$ to $\mathcal{P}$.
     If $phase = 1$, then $\mathcal{P}$ proceeds as follows.
     (a) $\mathcal{P}$ selects a random pairwise independent hash function $H_{k'}$ from the family $\mathcal{H}$, and sends $(H_{k'}, b \oplus H_{k'}(\boldsymbol{\sigma}))$ to $\mathcal{V}$.
     (b) $\mathcal{P}$ and $\mathcal{V}$ both output $\left(\mathbf{c}^{(1)}, F_k, \mathbf{y}, phase = 1, H_{k'}, b \oplus H_{k'}(\boldsymbol{\sigma})\right)$ as the commitment.

     If $phase = 2$, then $\mathcal{P}$ proceeds as follows.
     (a) $\mathcal{P}$ runs P_r^1 to obtain the decommitment message $\boldsymbol{\gamma}^{(1)}$ and first-phase transcript $\boldsymbol{\tau}$ corresponding to both $\boldsymbol{\sigma}$ and $\mathbf{c}^{(1)}$. $\mathcal{P}$ sends $(\boldsymbol{\sigma}, \boldsymbol{\gamma}^{(1)}, \boldsymbol{\tau})$ to $\mathcal{V}$.
     (b) $\mathcal{P}$ and $\mathcal{V}$ engage in $(P_c^2(b), V_c^2)(1^n, \boldsymbol{\tau})$, with $\mathcal{P}$ acting as P_c^2 and $\mathcal{V}$ acting as V_c^2.
     Let $\mathbf{c}^{(2)}$ be the common output of P_c^2, V_c^2 after the interaction.
     (c) $\mathcal{P}$ and $\mathcal{V}$ both output $\left(\mathbf{c}^{(1)}, F_k, \mathbf{y}, phase = 2, \mathbf{c}^{(2)}\right)$ as the commitment.

  Reveal phase: To open the commitment, $\mathcal{P}$ does the following depending on the value of $phase$.

  - If $phase = 1$, then $\mathcal{P}$ sends $b, \boldsymbol{\sigma}$ to $\mathcal{V}$.

- If $phase = 2$, then $\mathcal{P}$ runs P_r^2 to obtain the decommitment message $\boldsymbol{\gamma}^{(2)}$, and sends $b, \boldsymbol{\gamma}^{(2)}$ to $\mathcal{V}$.

- The algorithm Ver works as follows.
  - If $phase = 1$, then $\mathcal{V}$ checks that $\mathbf{y} = F_k(\boldsymbol{\sigma})$, and verifies that the last component of the commitment equals $b \oplus H_{k'}(\boldsymbol{\sigma})$.
  - If $phase = 2$, then $\mathcal{V}$ checks that $\mathbf{y} = F_k(\boldsymbol{\sigma})$, and verifies that both V_r^1 and V_r^2 accept $\mathbf{c}^{(1)}, \boldsymbol{\sigma}, \boldsymbol{\gamma}^{(1)}$, and $\mathbf{c}^{(2)}, b, \boldsymbol{\gamma}^{(2)}$, respectively.

[a] As mentioned before, the 1-out-of-2 binding property allows $\mathcal{P}$ to cheat in one of both stages. However, the universal one-way hash function F_k forces $\mathcal{P}$ to decide in which stage it likes to cheat before knowing the value of *phase*.

According to [HNO+09], applying Theorem 2.3 to each two-phase commitment scheme obtained from Theorem 2.2, yields a collection of public coin (standard) commitment schemes $\{(\text{Setup,Com,Ver})_i\}_{i=1}^m$ such that the following hold.

- There exists an index $i \in [m]$ such that the scheme $(\text{Setup,Com,Ver})_i$ is statistically hiding.
- For every index $i \in [m]$, the scheme $(\text{Setup,Com,Ver})_i$ is computationally binding.

Finally, the latter collection of (standard) commitment schemes can be converted, as stated in [HNO+09], into the desired statistically hiding commitment scheme by randomly secret-sharing the bit $b = b_1 \oplus \ldots \oplus b_m$, where b is the bit being committed to, and committing to each share b_i using the i'th commitment scheme $(\text{Setup,Com,Ver})_i$, for all $i \in [m]$.

Remark 2.2. The work of [HHK+05] shows how to construct a statistically hiding commitment scheme given any *regular* one-way function with known preimage size, i.e., a OWF f, for which every point in the image of f has the same number of preimages, and this number is known or it can be efficiently computed. More generally, a construction of statistically hiding commitment scheme was proposed in [HHK+05], which is based on *approximable-preimage-size* OWF, i.e., a OWF f satisfying the additional property that given any image y of f, the number of points mapping to y can be efficiently approximated. However, A construction of statistically hiding commitment scheme based on any arbitrary, unstructured OWF was left by [HHK+05] as an open problem. The contribution of [HNO+09], which we reviewed above, resolved this open problem.

2.2 Commitments from One-Way Permutations

The constructions we review in this section were presented in [GK96, NOVY98, TPY08]. They all assume the existence of one-way permutations (OWPs). As we will see, OWPs can be used to construct commitment schemes that are either perfectly hiding or perfectly binding.

2.2.1 Perfectly Binding from [GK96]

Hereafter we review a non-interactive perfectly binding commitment scheme for committing to single bits. The construction was proposed in [GK96], and uses the *Goldreich-Levin Theorem* introduced in [GL89], which essentially gives a reduction from inverting a one-way function to predicting a particular hard-core bit associated with that function. Loosely speaking, a hard-core predicate of a function is a bit that can be efficiently computed from the input to the function and yet is hard to predict from the output of the function, i.e., the probability of predicting the bit is not significantly better than one-half. Hard-core predicates were introduced in [BM84]. A formal definition follows.

Definition 2.4 (Hard-Core Predicate)**.** A polynomial-time computable predicate $h\colon \{0,1\}^* \longrightarrow \{0,1\}$ is called a *hard-core* of a function $f\colon \{0,1\}^* \longrightarrow \{0,1\}^*$, if for every probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function ε such that

$$\operatorname{prob}\big[\mathcal{A}(1^n, f(\mathbf{x})) = h(\mathbf{x})\big] \leq \frac{1}{2} + \varepsilon(n),$$

where the probability is taken over the uniform choice of $\mathbf{x} \in \{0,1\}^n$ and the random coin tosses of $\mathcal{A}$.

Next we recall the Goldreich-Levin Theorem from [GL89] that gives a hard-core predicate for a special form of OWFs.

Theorem 2.4 (Goldreich-Levin)**.** *Let $f\colon \{0,1\}^n \longrightarrow \{0,1\}^n$ be an arbitrary one-way permutation, and let $g\colon \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n \times \{0,1\}^n$ be the permutation defined by $g(\mathbf{x},\mathbf{r}) = (f(\mathbf{x}),\mathbf{r})$. Let $h(\mathbf{x},\mathbf{r}) = \langle \mathbf{x},\mathbf{r}\rangle$. Then the predicate h is a hard-core of the function g.*

In other words, the theorem states that if f is one-way permutation (OWP), then it is infeasible to guess the XOR of a random subset of the bits of $\mathbf{x}$ when given $f(\mathbf{x})$ and the subset itself.

Remark 2.3. A simple reduction proof shows that if f is OWP, then the function g defined in Theorem 2.4 is OWP as well.

The construction in [GK96] uses the Goldreich-Levin Theorem to obtain a non-interactive perfectly binding bit commitment scheme from any OWP with its associated hard-core predicate defined in the theorem.

- Setup(1^n): The setup algorithm selects any OWP $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ and the corresponding hard-core predicate h defined in the Goldreich-Levin Theorem 2.4.
- Com(f, h, b): The algorithm Com works as follows.
  Commit phase: To commit to a bit b. The prover $\mathcal{P}$ picks random n-bit strings $\mathbf{x}, \mathbf{r}$, and computes $c = b \oplus h(\mathbf{x}, \mathbf{r})$. Thereafter, $\mathcal{P}$ computes $\mathbf{y} = f(\mathbf{x})$ and sends the commitment $(\mathbf{y}, \mathbf{r}, c)$ to the verifier $\mathcal{V}$.
  Reveal phase: To decommit b, $\mathcal{P}$ sends $\mathbf{x}$ to $\mathcal{V}$.
- Ver($f, h, \mathbf{y}, \mathbf{r}, c, \mathbf{x}$): $\mathcal{V}$ verifies that $f(\mathbf{x}) = \mathbf{y}$, and then computes $b = c \oplus h(\mathbf{x}, \mathbf{r})$.

Since f is OWP, there is at most one value $\mathbf{x}$, which is a valid opening for the committed bit b. Thus, the scheme is perfectly binding. Moreover, Theorem 2.4 implies that the scheme is computationally hiding, since any probabilistic polynomial-time algorithm $\mathcal{P}$, which enables $\mathcal{V}$ to guess the bit b from the commitment $(\mathbf{y}, \mathbf{r}, c)$ with probability significantly better than one-half, can be converted to an algorithm that inverts f with non-negligible probability, contradicting the one-wayness of f.

Remark 2.4. The above described construction proposed in [GK96] was also appeared in [AC02], where the Goldreich-Levin Theorem 2.4 was investigated in the context of quantum information. While in [AC02], the authors described the construction as given above, the work of [GK96] described the construction in general terms. More precisely, the work of [GK96] assumed without loss of generality that the one-way permutation f has a hard-core predicate h. Thus, the commitment of a bit b is the string $(f(\mathbf{x}), h(\mathbf{x}) \oplus b)$, for a random bit string $\mathbf{x} \in \{0,1\}^n$.

2.2.2 Perfectly Hiding from [NOVY98]

This construction provides an interactive perfectly hiding commitment scheme that commits to single bits. The construction was presented in [NOVY98] and employs a technique called *interactive hashing* as a subroutine.

Interactive hashing was introduced in [OVY91]. It is a cryptographic protocol that allows two parties, a sender $\mathcal{S}$ and a receiver $\mathcal{R}$, to interact. During the interaction, $\mathcal{S}$ transfers a private input s to $\mathcal{R}$ such that $\mathcal{R}$ obtains two outputs, labeled s_0, s_1 according to lexicographic order.

For a formal definition of an interactive hashing protocol, we refer to [OVY91] or [HNO+09]. Instead, we informally recall the following two properties of such a protocol which are sufficient for our purposes.

1. **Binding**: Provided that typically both outputs are also available to the sender, the protocol guarantees that one of both outputs s_0 or s_1 is equal to the original input s, which the sender can control. The other string is uniformly distributed in the sense that it is chosen beyond the sender's control, even if it acts dishonestly.

2. **Hiding**: Provided that the sender's private input s is uniformly random, the protocol guarantees with probability more than one-half that the receiver cannot guess which of the two strings s_0, s_1 was the original input, even if the receiver is a computationally unbounded malicious one.

Next we review the construction from [NOVY98].

- Setup(1^n): For a security parameter $n \in \mathbb{N}$, the setup algorithm selects any one-way permutation $f : \{0,1\}^n \longrightarrow \{0,1\}^n$.

- Com(f, b): The algorithm Com works as follows.
  Commit phase: To commit to a bit $b \in \{0,1\}$, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ act as follows.

  1. $\mathcal{P}$ selects a bit string $\mathbf{r} \in \{0,1\}^n$ uniformly at random, and computes $\mathbf{y} = f(\mathbf{r})$.
  2. $\mathcal{P}$ and $\mathcal{V}$ engage in the following interactive hashing protocol.
     (a) $\mathcal{V}$ selects elements $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{n-1} \in \{0,1\}^n$ such that each $\mathbf{h}_i$, for $i \in [n-1]$, is a random vector over $\mathbf{GF}(2)$ of the form $0^{i-1}\|1\|\{0,1\}^{n-i}$, i.e., $i-1$ zeros followed by a one followed by an arbitrary choice for the last $n-i$ entries [a].
     (b) For j from 1 up to $n-1$
     – $\mathcal{V}$ sends $\mathbf{h}_j$ to $\mathcal{P}$.
     – $\mathcal{P}$ sends $c_j \leftarrow \langle \mathbf{h}_j, \mathbf{y} \rangle$.

(c) At this point there are exactly two vectors $\mathbf{y}_0, \mathbf{y}_1 \in \{0,1\}^n$ such that $c_j = \langle \mathbf{h}_j, \mathbf{y}_i \rangle$, for all $1 \leq j \leq n-1$, and for both $i \in \{0,1\}$. Let $\mathbf{y}_0$ be the lexicographically smaller of the two vectors. Both $\mathcal{P}$ and $\mathcal{V}$ compute $\mathbf{y}_0$ and $\mathbf{y}_1$ by solving the linear system. Let $d \in \{0,1\}$ be such that $\mathbf{y} = \mathbf{y}_d$.

3. $\mathcal{P}$ sends the commitment $c = b \oplus d$ to $\mathcal{V}$.

Reveal phase: To open the commitment, $\mathcal{P}$ sends b and $\mathbf{r}$ to $\mathcal{V}$.

- $\mathrm{Ver}(f, c, \mathbf{h}_1, \ldots, \mathbf{h}_{n-1}, c_1, \ldots, c_{n-1}, b, \mathbf{r})$: $\mathcal{V}$ verifies that $f(\mathbf{r}) = \mathbf{y}$ satisfies $c_j = \langle \mathbf{h}_j, \mathbf{y} \rangle$, for all $1 \leq j \leq n-1$, and checks that $\mathbf{y}_d = \mathbf{y}$, where $d = c \oplus b$.

[a] The vectors $\mathbf{h}_1, \ldots, \mathbf{h}_{n-1}$ are linearly independent over the Galois field $\mathbf{GF}(2)$.

We provide an illustration of the protocol flow in Figure 2.4.

Since the bit string $\mathbf{r}$ is chosen uniformly at random and since f is one-way permutation (OWP), then the bit string $\mathbf{y}$ is uniformly random as well. Moreover, by the hiding property of the interactive hashing protocol, the verifier $\mathcal{V}$ does not know if $\mathbf{y} = \mathbf{y}_0$ or $\mathbf{y} = \mathbf{y}_1$, since both $\mathbf{y}_0$ and $\mathbf{y}_1$ are uniformly distributed, even if $\mathcal{V}$ is computationally unbounded malicious. Consequently, $\mathcal{V}$ does not know if $d = 0$ or $d = 1$. It follows the perfectly hiding property. On the other hand, by the binding property of the interactive hashing protocol, at least one of the outputs, say $\mathbf{y}_a$, for $a \in \{0,1\}$, is uniform in $\{0,1\}^n$ and outside the prover's control. Thus, if the prover is able to decommit to both $b = 0$ and $b = 1$, it must find a preimage of $\mathbf{y}_a$. This task is computationally infeasible since f is OWP. It follows the computationally binding property.

2.2.3 Perfectly Hiding from [TPY08]

In the following we review a construction that provides an interactive perfectly hiding commitment scheme with two rounds. The protocol was proposed in [TPY08] and commits to single bits.

- $\mathrm{Setup}(1^n)$: For a security parameter $n \in \mathbb{N}$, the setup algorithm selects any one-way permutation $f : \{0,1\}^n \longrightarrow \{0,1\}^n$.
- $\mathrm{Com}(f, b)$: The algorithm Com works as follows.
  Commit phase: To commit to a bit $b \in \{0,1\}$, the prover $\mathcal{P}$ and the

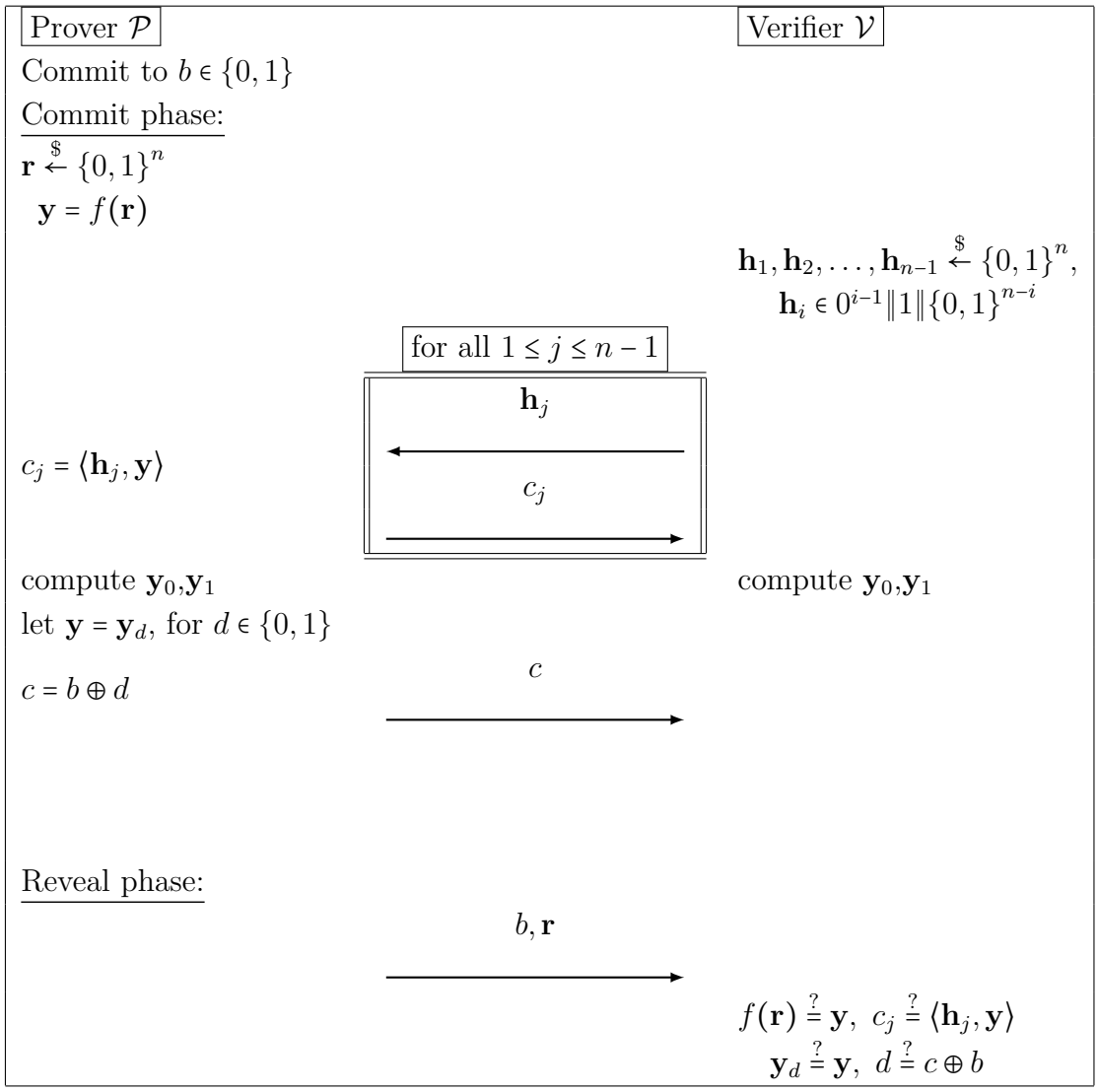

Figure 2.4: Perfectly hiding bit commitment protocol from [NOVY98].

verifier $\mathcal{V}$ act as follows.

1. $\mathcal{V}$ picks a uniformly random bit string $\mathbf{r} \in \{0,1\}^n$ and sends the string $\mathbf{r}$ to $\mathcal{P}$.
2. Upon receiving $\mathbf{r}$, $\mathcal{P}$ selects a bit string $\mathbf{x} \in \{0,1\}^n$ uniformly at random and sends the commitment string $\mathbf{c}$ to $\mathcal{V}$, where

$$\mathbf{c} = \begin{cases} f(\mathbf{x}), & \text{if } b = 0 \\ f(\mathbf{x}) \oplus \mathbf{r}, & \text{if } b = 1 \ . \end{cases}$$

Reveal phase: To open the commitment, $\mathcal{P}$ sends the string $\mathbf{x}$ to $\mathcal{V}$.

- $\mathrm{Ver}(f, \mathbf{r}, \mathbf{c}, \mathbf{x})$: $\mathcal{V}$ extracts the bit b, where

$$b = \begin{cases} 0, & \text{if } f(\mathbf{x}) = \mathbf{c} \\ 1, & \text{if } f(\mathbf{x}) \oplus \mathbf{r} = \mathbf{c} \end{cases}.$$

We illustrate the protocol flow in Figure 2.5.

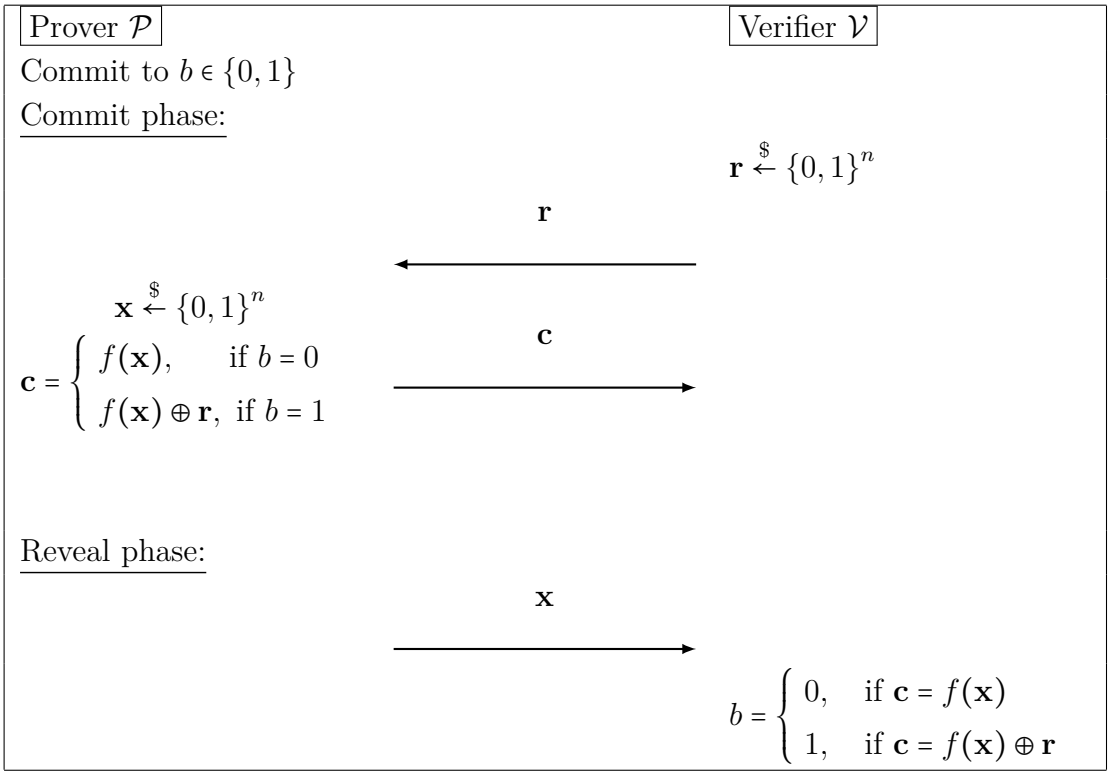

Figure 2.5: Perfectly hiding bit commitment protocol from [TPY08].

The computationally binding property follows from the fact that valid openings for both cases $b = 0$ and $b = 1$ with uniformly random bit strings $\mathbf{x}_0$ and $\mathbf{x}_1$, respectively, require that $f(\mathbf{x}_0) = f(\mathbf{x}_1) \oplus \mathbf{r}$. However, the prover holds such bit strings with negligible probability from f, since the strings $\mathbf{r}, f(\mathbf{x})$, and $f(\mathbf{x}) \oplus \mathbf{r}$ are equally distributed. The latter holds since f is OWP. On the other hand, inverting the function f by any verifier, even a computationally unbounded, opens both cases $b = 0$ and $b = 1$ with the same probability, since it can find any unique $\mathbf{x}_0$ such that $f(\mathbf{x}_0) = \mathbf{c}$ and unique $\mathbf{x}_1$ such that $\mathbf{c} = f(\mathbf{x}_1) \oplus \mathbf{r}$ from f. Hence, any verifier can correctly guess the committed bit from the commitment string with probability exactly one-half. It follows the perfectly hiding property.

2.3 Commitments from Collision-Free Hashing

This section reviews constructions of commitment schemes from general frameworks that assume the existence of collision resistant hash functions (CRHFs). These frameworks were proposed in [HM96, DPP97, DPP98]. They all provide statistically hiding commitments. Beside these frameworks, we also review constructions of CRHFs presented in [IKO05], which are based on perfectly binding commitments that hold additional properties. Consequently, we obtain in turn statistically hiding commitments.

2.3.1 Statistically Hiding from [HM96]

The commitment scheme we review hereafter was presented in [HM96]. The protocol is non-interactive, commits to bit strings, and is statistically hiding. Additionally, it uses *2-universal hash functions*, introduced in [CW77], as a tool for adding randomness to the message being committed to. A formal definition follows.

Definition 2.5 (2-Universal Hash Functions)**.** A family of hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$ from a domain X to a range Y with security parameter $n \in \mathbb{N}$ is called *2-universal*, if for every two distinct domain elements $x_1, x_2 \in X$ it holds that

$$\mathrm{prob}\left[k \leftarrow \mathrm{KGen}(1^n) \ : \ H_k(x_1) = H_k(x_2)\right] \leq \frac{1}{|Y|} \ .$$

We let n denote the security parameter of the scheme. The security parameter controls the success probability of the prover in changing the message after the commit phase as well as the probabilistic advantage the verifier may get about the message from its commitment. Furthermore, the construction sets a parameter t to be a positive integer at least $4n + 2\ell + 4$, where ℓ is the message length.

- Setup(1^n): The setup algorithm selects any collision-free hash function f such that $f : \{0,1\}^t \longrightarrow \{0,1\}^n$ and any family of 2-universal hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$ from $\{0,1\}^t$ to $\{0,1\}^\ell$.
- Com($f, \mathcal{H}, \mathbf{m}$): The algorithm Com works as follows.
  Commit phase: To commit to a message $\mathbf{m} \in \{0,1\}^\ell$, the prover $\mathcal{P}$ selects a random bit string $\mathbf{r} \in \{0,1\}^t$ and computes $\mathbf{y} = f(\mathbf{r})$. Moreover, $\mathcal{P}$ picks a random function H_k from the family $\mathcal{H}$ for which $H_k(\mathbf{r}) = \mathbf{m}$. The commitment is the string $\mathbf{c} = (H_k, \mathbf{y})$.

Reveal phase: Opening the commitment is performed by sending the random string $\mathbf{r}$.

- $\mathrm{Ver}(f, \mathcal{H}, \mathbf{c}, \mathbf{r})$: The verifier $\mathcal{V}$ checks that $\mathbf{y} = f(\mathbf{r})$ and then computes $\mathbf{m} = H_k(\mathbf{r})$.

The selection of a random function H_k from the family of 2-universal hash functions $\mathcal{H}$ such that $H_k(\mathbf{r}) = \mathbf{m}$, depends on the structure of the chosen family. The following example gives a family of 2-universal hash functions suggested in [HM96].

Example 2.1 (2-Universal Hashing Construction I)**.** For $t > \ell$, consider the family $\mathcal{H} = \{H_{\mathbf{A},\mathbf{b}} : \mathbf{A} \in \{0,1\}^{\ell \times t}, \mathbf{b} \in \{0,1\}^{\ell}\}$, where $\mathbf{A} = (a_{i,j})$ is a Toeplitz matrix, i.e., a matrix with all diagonals being homogeneous: $a_{i,j} = a_{i+1,j+1}$, for all i, j. Each pair $(\mathbf{A}, \mathbf{b})$ describes a function $H_{\mathbf{A},\mathbf{b}} : \{0,1\}^t \longrightarrow \{0,1\}^{\ell}$ such that $H_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, for all $\mathbf{x} \in \{0,1\}^t$, where the operations are taken over the field $\mathbf{GF}(2)$. Any function from this family can be described using $t + 2\ell - 1$ bits.

If the family of 2-universal hash functions given in Example 2.1 is used for the commitment scheme described above, a random function $H_{\mathbf{A},\mathbf{b}}$ has to be selected such that $H_{\mathbf{A},\mathbf{b}}(\mathbf{r}) = \mathbf{m}$. To do this, the matrix $\mathbf{A}$ is first chosen randomly from $\{0,1\}^{\ell \times t}$ and then the vector $\mathbf{b}$ is computed such that $\mathbf{b} = \mathbf{m} - \mathbf{A}\mathbf{r} \in \{0,1\}^{\ell}$.

For analyzing the commitment scheme, it is clear that opening the commitment for two different messages $\mathbf{m}, \mathbf{m}'$ implies finding a collision in f, i.e., finding $\mathbf{r}'$ such that $\mathbf{y} = f(\mathbf{r}')$. It follows that the scheme is computationally binding. On the other hand, applying a collision-free hash function on a random string $\mathbf{r}$ leaks no information about the committed message $\mathbf{m}$. Furthermore, using a 2-universal hash function produces randomness to the message, so that the verifier gets almost no statistical advantage about the message from the commitment string $\mathbf{c}$. It follows that for any two messages $\mathbf{m}_1, \mathbf{m}_2$, the distribution $(f, \mathcal{H}, \mathbf{c}_1)$ and $(f, \mathcal{H}, \mathbf{c}_2)$ are statistically close, where $\mathbf{c}_1, \mathbf{c}_2$ are the commitment strings of the messages $\mathbf{m}_1, \mathbf{m}_2$, respectively. Hence, the scheme is statistically hiding.

As proposed in [HM96], the communication complexity of the commitment scheme described above can be reduced by committing to the hash value $\mathbf{s} = f(\mathbf{m})$ of the message $\mathbf{m}$ instead of the message itself. To decommit $\mathbf{m}$, the prover sends $\mathbf{m}$ along with the random string $\mathbf{r}$. The verifier checks that $\mathbf{s}$ is the string being committed to, and that $f(\mathbf{m}) = \mathbf{s}$. Consequently, the commitment and decommitment strings are of length $\ell + O(n)$ instead of $O(\ell + n)$ bits.

Remark 2.5. A simple non-interactive string commitment scheme was suggested

in [Ste96]. The scheme uses some collision resistant hash function f as well. The commitment of a bit string $\mathbf{m}$ is the string $f(\mathbf{r} \| (\mathbf{r} \oplus \mathbf{m}))$, for a random bit string $\mathbf{r}$. The computationally binding property follows directly from the collision resistance of f. However, its hiding property was not proven.

2.3.2 Statistically Hiding from [DPP97]

Hereafter we review a construction presented in [DPP97], which transforms any *fail-stop signature* (FSS) scheme with a property called *almost unique secret key* into an interactive statistically hiding bit commitment scheme. FSS schemes were introduced in [WP89], and can be constructed from any collision resistant hash function, as shown in [DPP97]. Hence, the transformation can be based on any collision resistant hash function.

FSS schemes hold additional properties that make them differ from ordinary signature schemes. For a formal definition of FSS schemes and their properties, we refer to [WP89, DPP97]. Instead, we follow [DPP97] to informally recall the description of the properties of FSS schemes, which are sufficient for our purposes.

- A FSS scheme has several possible secret keys corresponding to a given public key. Any adversary, even computationally unbounded, cannot guess from publicly available information which of the possible secret keys is known to the signer. Consequently, it is impossible to predict which signature the signer would produce on a given message, if it has not been signed yet. This is because usage of different secret keys in general leads to different signatures.

- From two different signatures on the same message, the signer can produce a *proof of forgery*. On the other hand, the signer cannot falsely repudiate its own signature, if it has in fact not been forged, unless the signer breaks the computational assumption on which the security of the FSS scheme is based.

- To ensure that the signer does not generate a key pair for which it can prove its own signatures to be forgeries, the key generation process must not carried out by the signer alone. Thus, it is a protocol (rather than an algorithm) executed by the signer and a center trusted by the recipients.

- Except with an extremely small probability, it is impossible, even for a computationally unbounded forger, to produce a signature, which the signer cannot prove to be a forgery.

- The almost unique secret key property means that it is infeasible for a signer to compute more than one significantly different secret key corresponding to its public key. Keys are not significantly different if they lead to equal signatures.

The main idea in the transformation of a FSS scheme with the almost unique secret key property into a bit commitment scheme, is to use the key generation protocol as a commitment protocol, where the verifier plays the role of the trusted center. The key generation protocol has two main security parameters n, κ, where n is the security parameter for the recipient, and κ is for the signer. The transformation uses some family of 2-universal hash functions (see Definition 2.5).

- Setup(1^n): The setup algorithm initializes the key generation protocol $\mathrm{KGen}(1^n, 1^\kappa)$ of some FSS scheme with the almost unique secret key property, where n is the security parameter of the commitment scheme and $\kappa = 4(n+1)$. The part of the secret key $\mathbf{sk}$ that makes differences in the signatures is denoted by $\mathbf{n(sk)}$. Moreover, the algorithm selects any family of 2-universal hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$ from $\{0,1\}^{p(n)}$ to $\{0,1\}$, where $p(n)$ is an upper bound on the length of the values of $\mathbf{n(sk)}$ for the parameters n, κ.

- $\mathrm{Com}(\mathrm{KGen}(1^n, 1^\kappa), \mathcal{H}, b)$: The algorithm Com works as follows.
  Commit phase: To commit to a bit $b \in \{0,1\}$, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ act as follows.

  1. $\mathcal{P}$ and $\mathcal{V}$ engage in the key generation protocol $\mathrm{KGen}(1^n, 1^\kappa)$. If $\mathcal{P}$ or $\mathcal{V}$ reject in the key generation, the commit phase stops. Let $\mathbf{pk}$ and $\mathbf{sk}$ be the resulting public and private key, respectively.
  2. $\mathcal{P}$ signs the bit 0, and generates a proof of forgery on the resulting signature. If this results in a valid proof of forgery, $\mathcal{P}$ stops. Otherwise $\mathcal{P}$ continues.
  3. $\mathcal{P}$ picks a random 2-universal hash function H_k from the family $\mathcal{H}$.
  4. $\mathcal{P}$ computes $c = H_k(\mathbf{n(sk)}) \oplus b$, and sends (H_k, c) to $\mathcal{V}$.

  Reveal phase: To open the commitment, $\mathcal{P}$ sends b and $\mathbf{sk}$ to $\mathcal{V}$.

- $\mathrm{Ver}(\mathrm{KGen}(1^n, 1^\kappa), \mathcal{H}, \mathbf{pk}, H_k, c, b, \mathbf{sk})$: $\mathcal{V}$ verifies that $\mathbf{sk}$ fits to $\mathbf{pk}$, and compares b with $c \oplus H_k(\mathbf{n(sk)})$.

We illustrate the protocol flow in Figure 2.6.

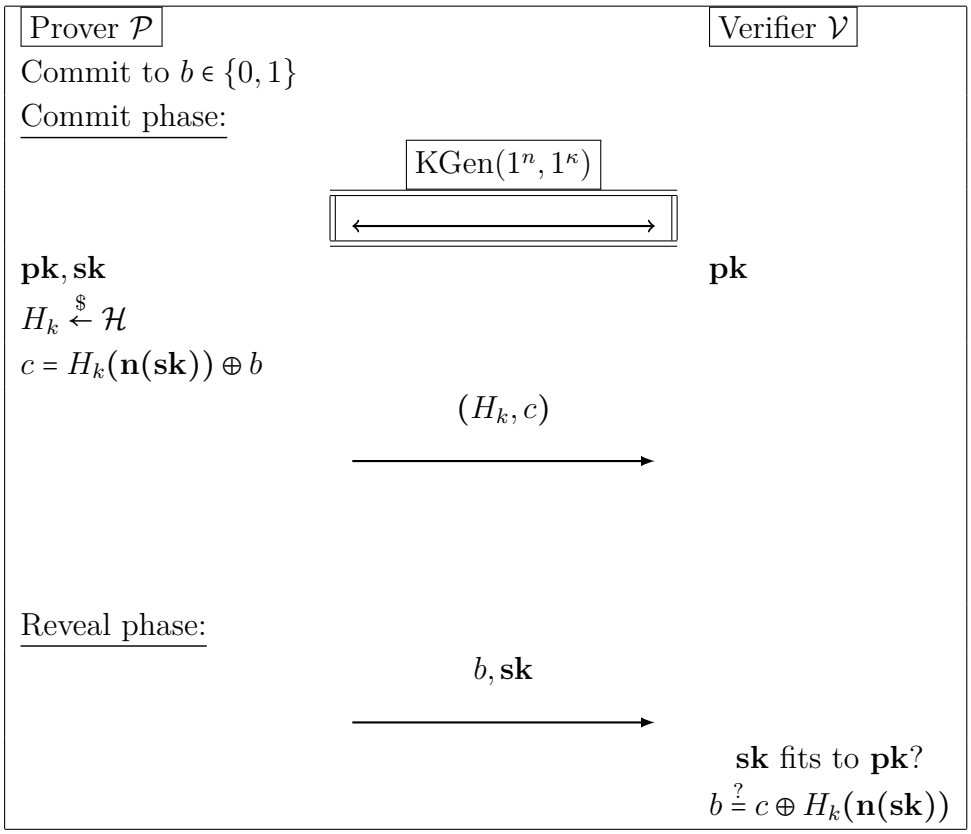

Figure 2.6: Statistically hiding bit commitment protocol from [DPP97].

The computationally binding property follows directly from the almost unique secret key property of the FSS scheme. Additionally, since the distribution of the secret key, given the public key, is not necessarily uniform, a family of 2-universal hash functions is used to add randomness to the commitment value. Moreover, FSS schemes do not exclude that a cheating center (here, the verifier) can carry out key generation so that it can guess the secret key afterward. Thus, the *extended privacy amplification Theorem*, introduced in [BBCM95], is used as well, to derive that the verifier has very little collision information about $H_k(\mathbf{n}(\mathbf{sk}))$, as the bit b is encrypted with the privacy-amplified significant part of $\mathbf{pk}$, i.e., $c = H_k(\mathbf{n}(\mathbf{sk})) \oplus b$. Hence, the family of 2-universal hash functions together with the extended privacy amplification Theorem achieve the statistically hiding property.

Remark 2.6. As stated in [DPP97], the security proof of the construction allows to commit to more than one bit, i.e., there is no need to hash $\mathbf{n}(\mathbf{sk})$ down to a 1-bit value to wipe out the verifier's information.

2.3.3 Statistically Hiding from [DPP98]

We review below a construction proposed in [DPP98] which provides a statistically hiding commitment scheme. The scheme is non-interactive, commits to bit strings, and uses some family of 2-universal hash functions (see Definition 2.5).

In order to achieve efficiency, a particular family of 2-universal hash functions was suggested in [DPP98]. The following example gives this particular family, which is essentially from [CW77].

Example 2.2 (2-Universal Hashing Construction II)**.** For $t > \ell$, consider the family $\mathcal{H} = \{H_{\mathbf{a},\mathbf{b}} : \mathbf{a}, \mathbf{b} \in \{0,1\}^t\}$. Each pair $(\mathbf{a}, \mathbf{b})$ describes a function $H_{\mathbf{a},\mathbf{b}} : \{0,1\}^t \longrightarrow \{0,1\}^\ell$ such that $H_{\mathbf{a},\mathbf{b}}(\mathbf{x}) = \mathbf{ax} + \mathbf{b}\big|_\ell$, for all bit strings $\mathbf{x} \in \{0,1\}^t$, where $\big|_\ell$ means taking the ℓ least significant bits from the output of the function. The operations are taken over the field $\mathbf{GF}(2)$. Any function from this family can be described using $2t$ bits.

We let n denote the security parameter, and ℓ the length of the message being committed to. Furthermore, the commitment scheme sets a parameter t to be a positive integer such that $t = 3(n+1)$.

- Setup(1^n): The setup algorithm selects any collision-free hash function f such that $f : \{0,1\}^* \longrightarrow \{0,1\}^{n+1}$, and any family of 2-universal hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$ from $\{0,1\}^t$ to $\{0,1\}^{n+1}$.
- Com($f, \mathcal{H}, \mathbf{m}$): The algorithm Com works as follows.
  Commit phase: To commit to a message $\mathbf{m} \in \{0,1\}^\ell$, the prover $\mathcal{P}$ selects a random bit string $\mathbf{r} \in \{0,1\}^t$, and a random function H_k from the family $\mathcal{H}$. The commitment is the hash value
  $$\mathbf{c} = f\Big(H_k \| f(\mathbf{r}) \| f(\mathbf{m}) \oplus H_k(\mathbf{r})\Big) .$$
  Reveal phase: Opening the commitment is performed by sending $H_k, \mathbf{r}$, and $\mathbf{m}$.
- Ver($f, \mathcal{H}, \mathbf{c}, H_k, \mathbf{r}, \mathbf{m}$): The verifier $\mathcal{V}$ checks that
  $$f\Big(H_k \| f(\mathbf{r}) \| f(\mathbf{m}) \oplus H_k(\mathbf{r})\Big) = \mathbf{c} .$$

The computationally binding property is an immediate consequence of the collision resistance of the function f. In order to analyze the statistically hiding property, we

follow the observation from [DPP98] which states that it is enough to show the hiding property for a modified commitment string in which $H_k, f(\mathbf{r})$, and $f(\mathbf{m}) \oplus H_k(\mathbf{r})$ are sent. This modification gives the verifier even more information than the original commitment string. The *extended privacy amplification Theorem*, introduced in [BBCM95], interprets the part $f(\mathbf{m}) \oplus H_k(\mathbf{r})$ as encrypting $f(\mathbf{m})$ with the privacy amplified version of $\mathbf{r}$. Applying this theorem gives a bound on the verifier's expected uncertainty about $H_k(\mathbf{r})$, if it knows f, H_k, and $f(\mathbf{r})$. This bound is at least $n + 1 - 2^{-n}$. Thus, the verifier has almost no information about the committed message.

As mentioned in [DPP98], among the three applications of the function f in the commit phase $f(\mathbf{r})$, $f(\mathbf{m})$, and $f\Big(H_k \| f(\mathbf{r}) \| f(\mathbf{m}) \oplus H_k(\mathbf{r})\Big)$, only $f(\mathbf{r})$ is essential for the security of the scheme.

In [DPP98], the authors stated that hashing the message $\mathbf{m}$ may be omitted if it is short. The final hashing of the commitment may be also omitted in applications where the efficiency of the reveal phase is more important than that of the commit phase. The above described version with very short commitments and longer revealing is particularly suitable if not all commitments are opened.

Remark 2.7. In [DPP98], the authors stated that their construction is an improvement of a commitment scheme suggested in [NY89]. This scheme employs the Lamport-Diffie one-time signature scheme (LD-OTS) proposed in [Lam79]. The LD-OTS scheme uses any collision resistant hash function to hash the message to be signed and then use any one-way function for signing the message. The resulting commitment scheme needs interaction between the prover and the verifier in an initialization phase (the setup phase) for each commitment, i.e., each time when the prover wants to commit to a message, a one-time public key has to be provided for both the prover and the verifier before the commit phase.

2.3.4 Sufficient Conditions for Collision-Free Hashing

Since the beginning of this section, we have seen that the existence of collision resistant hash functions (CRHFs) implies commitment schemes that are statistically hiding. Essentially, CRHFs are important cryptographic primitive that can be constructed from general assumptions. For example, there are two constructions of CRHFs presented in [IKO05] that are related to commitments. Namely, they used commitment schemes with additional properties to obtain CRHFs, which in turn imply statistically hiding commitments, as seen in this section. Therefore, we review here these two constructions.

The first construction of CRHFs is based on *homomorphic encryption*. Roughly speaking, a homomorphic encryption scheme is, according to [IKO05], a semantically secure encryption scheme in which the plaintexts are taken from some group G, and given encryptions c_1, c_2 of two group elements m_1, m_2, it is possible to efficiently compute an encryption of $m_1 \boxplus m_2$ from c_1 and c_2, where $\boxplus$ is the operation defined on the group. Usually, a group G with an operation $\boxplus$ defined on it, is written as $(G, \boxplus)$.

Since the decryption algorithm is not used in the construction of CRHFs, the authors in [IKO05] reduced the construction to a weaker assumption. Namely, to the existence of *homomorphic commitments*. Loosely speaking, a homomorphic commitment scheme is, according to [IKO05], a semantically secure, perfectly binding, non-interactive commitment scheme with homomorphic property, i.e., from commitments c, c' of two messages m, m' from some group $(G, \boxplus)$, it is possible to efficiently compute a commitment to $m \boxplus m'$ from c and c'. A formal definition follows according to [IKO05].

Definition 2.6 (Homomorphic Commitment Scheme)**.** Let $(G, \boxplus)$ be some group of size $\alpha = poly(n)$, for a security parameter $n \in \mathbb{N}$. A *homomorphic commitment scheme* consists of a triple of probabilistic polynomial-time algorithms (KGen,Com,Eval) defined as follows.

- KGen(1^n): Given 1^n, output a public key pk.
- Com(pk, m): Given the public key pk, and a group element $m \in G$, output the value c of some length $\alpha' = poly(n)$. The commitment c hides the group element m, i.e., given c, the message m is semantically secure. Moreover, the commitment c is perfectly binding, i.e., given the public key pk, the commitment c uniquely determines m.

  Remark 2.8. The notation Com(pk, m) hides the fact that the algorithm Com is probabilistic. A deterministic value can be obtained by fixing some randomness r to the algorithm. In this case, we use the notation Com$(pk, m; r)$.

- Eval(pk, c, c'): Given the public key pk, and commitments Com(pk, m),Com(pk, m') of two group elements $m, m' \in G$, compute a commitment to $m \boxplus m'$, i.e., Com$(pk, m \boxplus m'; r)$, where r is any possible randomness for the algorithm Com. For any $t \geq 0$ and randomness r, the algorithm Eval guarantees that a commitment Com$(pk, tm; r)$ can be efficiently computed from Com$(pk, m; r)$ as well. This can be performed by applying the algorithm Eval $O(\log t)$ times.

The second construction of CRHFs uses *homomorphic one-way commitments* defined in [IKO05]. Homomorphic one-way commitments do not provide semantic security for committed values, i.e., they do not guarantee that no information about some committed value m is leaked. However, they guarantee that it is hard to find m from its commitment. In other words, only one-wayness security is required. A formal definition follows according to [IKO05].

Definition 2.7 (Homomorphic One-Way Commitment Scheme)**.** Let $(G, \boxplus)$ be some group of size $\alpha = poly(n)$, for a security parameter $n \in \mathbb{N}$. A *homomorphic one-way commitment scheme* is defined as a homomorphic commitment scheme (see Definition 2.6), except for the security requirement *one-wayness* defined as follows.

- Every probabilistic polynomial-time algorithm $\mathcal{A}$ that is given $\mathrm{Com}(pk, m; r)$ has a negligible probability of finding m, where the probability is taken over a random choice of $m \in G$, the choice of r by the algorithm Com, and the internal random choices of the algorithm $\mathcal{A}$.

  Remark 2.9. The perfectly binding property implies that there is a unique preimage m of the value $\mathrm{Com}(pk, m; r)$.

The following two remarks are concerning Definition 2.7.

Remark 2.10. Commitments are required to be re-randomized, i.e., there is a probabilistic polynomial-time algorithm such that given any commitment $\mathrm{Com}(pk, m; r)$, it outputs a re-randomized commitment.

Remark 2.11. The one-wayness requirement in particular implies that the group size α needs to be large.

Next we give both constructions of CRHFs shown in [IKO05].

I. CRHFs from homomorphic commitments

Given an arbitrary homomorphic commitment scheme (KGen′,Com,Eval) on some group $(G, \boxplus)$ of size $\alpha = poly(n)$, and commitment length $\alpha' = poly(n)$ with security parameter n, the construction of a family of CRHFs $\mathcal{F}_n$ =(KGen,Hash) is given as follows.

- KGen(1^n): Given 1^n, proceed as follows.

  1. Run the algorithm KGen$'(1^n)$ to obtain a public key pk' for the homomorphic commitment scheme.
  2. Choose a random $n_1 \times n_2$ matrix $\mathbf{M}$, whose entries are elements from G, where $n_1 = \lceil \frac{n}{\lfloor \log \alpha \rfloor} \rceil$, and n_2 is chosen such that $n_1 \alpha' < n_2 \log \alpha$ [a]. If the group size α is sufficiently large, then the value n_1 might be as small as 1.
  3. Run the algorithm Com$(pk', \mathbf{M})$, i.e., commit to each of the $n_1 \cdot n_2$ elements of the matrix $\mathbf{M}$.
  4. Finally, output $k = (pk', \text{Com}(pk', \mathbf{M}))$ as a key for the function.

- Hash$(k, \mathbf{x})$: Given the key k, and a message $\mathbf{x} = (x_1, \ldots, x_{n_2})$, where $x_i \in G, |x_i| = \lfloor \log \alpha \rfloor$, for all $i \in [n_2]$, output the digest

$$F_k(\mathbf{x}) = \text{Com}(pk', \mathbf{M} \cdot \mathbf{x}; r),$$

  where the commitment to $\mathbf{M} \cdot \mathbf{x}$ can be efficiently computed from k and $\mathbf{x}$ using the algorithm Eval (without knowledge of the matrix $\mathbf{M}$), and r is the randomness implicitly defined by this computation.

[a] This condition provides the compression property of the hash functions, i.e., the length of the output is required to be shorter than the length of its input.

The collision resistance property follows directly from the perfectly binding property of the homomorphic commitments in addition to the semantic security of the homomorphic commitment scheme.

II. CRHFs from homomorphic one-way commitments

Given an arbitrary homomorphic one-way commitment scheme (KGen$'$,Com,Eval) on some group $(G, \boxplus)$ of size $\alpha = poly(n)$, and commitment length $\alpha' = poly(n)$ with security parameter n, the construction of a family of CRHFs $\hat{\mathcal{F}}_n$ =(KGen,Hash) is given as follows.

- KGen(1^n): Given 1^n, proceed as follows.

  1. Run the algorithm KGen$'(1^n)$ to obtain a public key pk' for the homomorphic one-way commitment scheme.

2. Choose ℓ elements $m_1, \ldots, m_\ell$ randomly from G, where ℓ is chosen such that $\alpha' < \ell \log \alpha$ [a].
3. Choose random strings $r_1, \ldots, r_\ell$ to be used by the commitment algorithm Com.
4. Run the algorithm $\mathrm{Com}(pk', m_i; r_i)$, for each $i \in [\ell]$.
5. Finally, output $k = \left(pk', \left\{\mathrm{Com}(pk', m_i; r_i)\right\}_{i=1}^{\ell}\right)$ as a key for the function.

- $\mathrm{Hash}(k, \mathbf{x})$: Given the key k, and a message $\mathbf{x} = (x_1, \ldots, x_\ell)$, where $x_i \in G, |x_i| = \lfloor \log \alpha \rfloor$, for all $i \in [\ell]$, output the digest

$$\hat{F}_k(\mathbf{x}) = \mathrm{Com}(pk', \sum_{i=1}^{\ell} m_i x_i; r),$$

where the commitment to $\sum_{i=1}^{\ell} m_i x_i$ can be efficiently computed from k and $\mathbf{x}$ using the algorithm Eval (without knowledge of $m_1, \ldots, m_\ell$), and r is the randomness implicitly defined by this computation.

[a] This condition provides the compression property of the hash functions, i.e., the output length is required to be shorter than the input length.

The collision resistance property follows directly from the perfectly binding property of the homomorphic one-way commitments in addition to the one-wayness property of the homomorphic one-way commitment scheme.

2.4 Commitments from Public-Key Cryptosystems

This section includes one construction of a commitment scheme introduced in [CD04], which uses public-key encryption.

2.4.1 Statistically Binding from [CD04]

The commitment scheme we review below is non-interactive, commits to bit strings, and is statistically binding. The scheme was presented in [CD04], and assumes the existence of public-key cryptosystems whose encryption and decryption functions commute, i.e., either public or private key can be used in either function. More precisely, for some public-key cryptosystem (KGen,Enc,Dec) with public key

pk, and secret key sk, the encryption and decryption functions must satisfy that Enc(pk,Enc($sk, \mathbf{m}$))= $\mathbf{m}$, for any message $\mathbf{m}$. In such cryptosystems, encrypting some message using the secret key essentially corresponds to digitally signing the message. Thus we can write Enc(pk,Sig($sk, \mathbf{m}$))= $\mathbf{m}$. For simplicity, we let Sig($\mathbf{m}$), Enc($\mathbf{m}$) denote signing and encrypting any message $\mathbf{m}$ using the private and public key, respectively.

- Setup(1^n): The setup algorithm generates a key pair (pk, sk) for some commutative public-key cryptosystem (KGen,Enc,Dec) with security parameter n, where pk, sk denote the public and private key, respectively. The public key pk[a] is set as a public commitment key, while the secret key sk is kept private for the prover $\mathcal{P}$.

- Com($pk, \mathbf{m}$): The algorithm Com works as follows.
  Commit phase: To commit to a bit string $\mathbf{m}$, $\mathcal{P}$ generates a pseudorandom bit string $\mathbf{r}$. Thereafter, using the private key sk, $\mathcal{P}$ signs (or encrypts) the bit string $\mathbf{Id}\|\mathbf{m}$ to obtain Sig($\mathbf{Id}\|\mathbf{m}$), where $\mathbf{Id}$ denotes the identifier of $\mathcal{P}$. Furthermore, $\mathcal{P}$ encrypts the bit string $\mathbf{Id}\|\mathbf{r}$ by using the public key pk to obtain Enc($\mathbf{Id}\|\mathbf{r}$). Finally, $\mathcal{P}$ sends the following commitment string to the verifier.

  $$\mathbf{c} = \text{Sig}(\mathbf{Id}\|\mathbf{m}) \oplus \text{Enc}(\mathbf{Id}\|\mathbf{r}) \ .$$

  Reveal phase: For opening the commitment, $\mathcal{P}$ reveals the string $\mathbf{r}$.

- Ver($pk, \mathbf{c}, \mathbf{r}$): The verifier $\mathcal{V}$ encrypts $\mathbf{Id}\|\mathbf{r}$ using pk to obtain Enc($\mathbf{Id}\|\mathbf{r}$), and computes $\mathbf{c} \oplus$ Enc($\mathbf{Id}\|\mathbf{r}$) to retrieve Sig($\mathbf{Id}\|\mathbf{m}$). Finally, $\mathcal{V}$ encrypts Sig($\mathbf{Id}\|\mathbf{m}$) using pk to obtain $\mathbf{Id}\|\mathbf{m}$, and takes the suffix $\mathbf{m}$ of $\mathbf{Id}\|\mathbf{m}$ as the value committed to.

[a] Public keys are assumed to be certified and publicly accessible.

The commitment string $\mathbf{c}$ can be viewed as encrypting the string Sig($\mathbf{Id}\|\mathbf{m}$) using a symmetric stream cipher with a pseudorandom key Enc($\mathbf{Id}\|\mathbf{r}$). Thus, hiding the string Sig($\mathbf{Id}\|\mathbf{m}$) computationally, depends on the pseudorandom string $\mathbf{r}$ being generated using a cryptographically pseudorandom generator. Moreover, opening the commitment for two different messages $\mathbf{m}, \mathbf{m'}$ requires that $\mathbf{c} \oplus$ Enc($\mathbf{Id}\|\mathbf{r'}$) = Sig($\mathbf{Id}\|\mathbf{m'}$), for a string $\mathbf{r'}$. This implies that $\mathbf{Id}\|\mathbf{r'}$ = Sig($\mathbf{c} \oplus$ Sig($\mathbf{Id}\|\mathbf{m'}$)). The probability that the prefix of Sig($\mathbf{c} \oplus$ Sig($\mathbf{Id}\|\mathbf{m'}$)) matches the identifier string $\mathbf{Id}$ is roughly $2^{-|\mathbf{Id}|}$, where $|\mathbf{Id}|$ is the bit length of $\mathbf{Id}$. Thus, it follows the statistically binding property.

Remark 2.12. In order to obtain a negligible probability for opening a commitment for two different messages, the identifier string **Id** has to satisfy $|\mathbf{Id}| \geq 32$, as suggested in [CD04].

2.5 Commitments from Different Protocols

In this section we review two frameworks for commitment schemes introduced in [Dam89, BIKM99]. Both frameworks employ a cryptographic protocol that has to be executed or simulated each time during the commit phase.

2.5.1 Perfectly Hiding from Σ-Protocols

We review hereafter a construction of the perfectly hiding commitment scheme presented in [Dam89] which uses ***Σ-protocols***. Σ-protocols were introduced in [Cra96] as an abstract concept. Many works including the one of [Dam89] used Σ-protocols before conceptually presenting them in [Cra96]. First, we recall the definition of Σ-protocols and their properties. We mostly follow [Dam02], with some details from [DFS04].

A Σ-protocol is defined on some relation $R = \{(x, w)\}$. The language defined by R is written as $L_R = \{x \mid \exists w : (x, w) \in R\}$. For $x \in L_R$, any w such that $(x, w) \in R$ is called a *witness* for $x \in L_R$. The length of any witness w is restricted to be at most $poly(|x|)$.

Remark 2.13. For some $(x, w) \in R$, x can be considered as an instance of some computational problem, and w as the solution to that instance.

Definition 2.8 (Σ-Protocols)**.** Let $R = \{(x, w)\}$ be some relation, and L_R be a language defined by R. A ***Σ-protocol*** for R is a three-round interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, where some $x \in L_R$ is a common input for $\mathcal{P}, \mathcal{V}$, and some w such that $(x, w) \in R$ is a private input for $\mathcal{P}$. Both $\mathcal{P}, \mathcal{V}$ are probabilistic polynomial-time algorithms which act as follows.

1. First, $\mathcal{P}$ sends some message c to $\mathcal{V}$.

2. Then, $\mathcal{V}$ is only required to send a random ℓ-bit string $\mathbf{m}$ as a challenge to $\mathcal{P}$, where $\ell = poly(|x|)$.

3. Finally, $\mathcal{P}$ sends an answer z back to $\mathcal{V}$ which decides to accept or reject based on the input and the exchanged data, i.e., $x, c, \mathbf{m}$, and z.

A Σ-protocol for some relation R requires the following properties.

- **Completeness**: If $\mathcal{P}, \mathcal{V}$ follow the protocol on input x and private input w to $\mathcal{P}$, where $(x, w) \in R$, then $\mathcal{V}$ always accepts.
- **Special soundness**: From any x and any pair $(c, \mathbf{m}, z)$,$(c, \mathbf{m}', z')$ of accepting conversations on input x, where $\mathbf{m} \neq \mathbf{m}'$, there exists a probabilistic polynomial-time algorithm $\mathcal{E}$ which can efficiently compute a witness w such that $(x, w) \in R$.
- **Special honest-verifier zero-knowledge**: There exists a probabilistic polynomial-time simulator $\mathcal{M}$, which on input x and a random $\mathbf{m}$, it outputs an accepting conversation of the form $(c, \mathbf{m}, z)$ with the same probability distribution as conversations between the honest $\mathcal{P}$ and $\mathcal{V}$ on input x.

A relation R is defined to be hard if it is possible to efficiently generate pairs (x, w) such that, when given only x, it is hard to find a witness w for x. A formal definition follows.

Definition 2.9 (Hard Relation)**.** A relation R is called *hard* if the following conditions are satisfied.

- There exists a probabilistic polynomial-time algorithm $\mathcal{G}$, called the *generator*, which on input 1^n, outputs a pair $(x, w) \in R$, where $|x| = n$.
- For all probabilistic polynomial-time algorithms $\mathcal{A}$, the following holds.

$$\text{prob}\left[(x, w) \leftarrow \mathcal{G}(1^n),\ w_{\mathcal{A}} \leftarrow \mathcal{A}(x) :\ (x, w_{\mathcal{A}}) \in R\right] \leq negl(n)\ .$$

  In other words, on input x generated from $\mathcal{G}$, the algorithm $\mathcal{A}$ produces a valid witness for x only with negligible probability.

Next we review the construction of a perfectly hiding commitment scheme shown in [Dam89]. The scheme commits to single bits. However, according to [Dam02], the scheme allows committing to bit strings. We adopt the description of [Dam02] to review the construction of the commitment scheme.

- Setup(1^n): The setup algorithm selects some hard relation R with generator $\mathcal{G}$, and Σ-protocol $\mathbf{\Sigma}$ for R. R is selected such that it is easy to verify membership in L_R, i.e., given x, it is easy to decide if there exists w such that $(x, w) \in R$.

  Furthermore, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ interact as follows.

  1. $\mathcal{V}$ runs the generator $\mathcal{G}$ to obtain $(x, w) \in R$ and sends x to $\mathcal{P}$.
  2. $\mathcal{P}$ checks that $x \in R_L$.

- Com($R_L, \mathbf{\Sigma}, x, \mathbf{m}$): The algorithm Com works as follows.
  Commit phase: To commit to an ℓ-bit string $\mathbf{m}$, $\mathcal{P}$ runs the simulator $\mathcal{M}$ of $\mathbf{\Sigma}$ on input x and $\mathbf{m}$ to obtain $(c, \mathbf{m}, z)$, and sends the commitment c to $\mathcal{V}$.
  Reveal phase: To open the commitment, $\mathcal{P}$ sends $\mathbf{m}$ and z to $\mathcal{V}$.

- Ver($R_L, \mathbf{\Sigma}, x, c, \mathbf{m}, z$): $\mathcal{V}$ checks that $(c, \mathbf{m}, z)$ is an accepting conversation with respect to x.

The computationally binding property follows from the fact that opening the commitment c to both $(\mathbf{m}, z)$ and $(\mathbf{m'}, z')$, for $\mathbf{m} \neq \mathbf{m'}$, implies having two accepting conversations $(c, \mathbf{m}, z)$ and $(c, \mathbf{m'}, z')$. By the special soundness property, this means efficiently computing the witness w. This contradicts the assumption that the relation R is hard. Moreover, since $x \in R_L$, the special honest-verifier zero-knowledge property implies that the simulation by $\mathcal{M}$ is perfect. Hence, the message c generated by $\mathcal{M}$ is independent of the challenge $\mathbf{m}$. It follows the perfectly hiding property.

2.5.2 Statistically Hiding from PIR Protocols

Finally, we review below a construction proposed in [BIKM99] which provides a statistically hiding commitment scheme. The scheme is interactive, commits to single bits, and employs *private information retrieval* (PIR) protocols introduced in [CGKS98].

Roughly speaking, a PIR protocol is an interactive protocol that allows a user to access some database stored on a server without revealing to the server which information has been accessed, i.e., the retrieved data remain oblivious for the server. The database is modeled as an n-bit string $\mathbf{x} = (x_1, \ldots, x_n)$. When a user wishes to privately retrieve an item x_i, for $i \in [n]$, the retrieval process is performed

such that the server cannot (in polynomial-time) gain any information about the index the client is interested in.

We basically follow [BIKM99] to recall the formal definition of PIR protocols and their properties.

Definition 2.10 (Private Information Retrieval (PIR))**.** A *private information retrieval* (PIR) protocol involves two parties: a server $\mathcal{S}$ which hosts a database modeled as an n-bit string $\mathbf{x} = (x_1, \ldots, x_n)$ and a user $\mathcal{U}$ who wants to retrieve a bit x_i from the database, where $i \in [n]$. The protocol consists of a triple of the following polynomial-time computational algorithms.

- **Query**: $\mathcal{U}$ acts first by selecting a random string $\mathbf{r}$, computing a query q, i.e., $q \leftarrow \text{Query}(1^n, i, \mathbf{r})$, and then sending q to $\mathcal{S}$.
- **Answer**: $\mathcal{S}$ responds with an answer s, i.e., $s \leftarrow \text{Answer}(1^n, \mathbf{x}, q)$.
- **Decode**: $\mathcal{U}$ computes the bit x_i, i.e., $x_i \leftarrow \text{Decode}(1^n, i, \mathbf{r}, s)$.

A PIR protocol requires the following properties.

- **Perfect correctness**: The user always computes the correct value of x_i, i.e., for every index $i \in [n]$, every random string $\mathbf{r}$, and every database $\mathbf{x} = (x_1, \ldots, x_n)$, it holds that

$$\text{prob}\Big[\text{Decode}\Big(1^n, i, \mathbf{r}, \text{Answer}\big(1^n, \mathbf{x}, \text{Query}(1^n, i, \mathbf{r})\big)\Big) = x_i\Big] = 1 \ .$$

- **Privacy**: The server cannot (in polynomial-time) gain information about the bit that the user tries to retrieve, i.e., for any probabilistic polynomial-time adversary $\mathcal{A}$, and any two sequences of indices $\{\mathbf{i}_n\}_{n=1}^{\infty}$ and $\{\mathbf{j}_n\}_{n=1}^{\infty}$, where $1 \le \mathbf{i}_n, \mathbf{j}_n \le n$, it holds that

$$\Big|\text{prob}\big[\mathcal{A}(1^n, \mathbf{i}_n, \text{Query}(1^n, \mathbf{i}_n, \cdot)) = 1\big] - \text{prob}\big[\mathcal{A}(1^n, \mathbf{j}_n, \text{Query}(1^n, \mathbf{j}_n, \cdot)) = 1\big]\Big| \le negl(n) \ .$$

  In other words, the distributions $\text{Query}(1^n, \mathbf{i}_n, \cdot)$ and $\text{Query}(1^n, \mathbf{j}_n, \cdot)$ are computationally indistinguishable.

Next we review the construction of a statistically hiding bit commitment protocol presented in [BIKM99]. The binding property of the scheme is weak, i.e., a prover can open a commitment ambiguously with probability at most $1 - 1/p(n)$, for some polynomial $p(n)$.

According to [BIKM99], the weak binding property can be strengthened by requiring the prover $\mathcal{P}$ to independently commit to the same bit b polynomially many times (e.g., n^2) and letting the verifier $\mathcal{V}$ to output a bit b' only if the same bit b' was successfully decommitted every time, otherwise reject. The sequential repetition yields strong binding property without compromising the statistically hiding property, i.e., $\mathcal{P}$ can cheat successfully only if it cheats successfully in each of the polynomially many repetitions, and this happens with exponentially small probability.

- Setup(1^n): The setup algorithm initializes a (multi-round) PIR protocol PIR=(Query,Answer,Decode), in which the server communicates at most $n/2$ bits.
- Com(PIR, b): The algorithm Com works as follows.
  Commit phase: To commit to a bit $b \in \{0,1\}$, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ act as follows.
  1. $\mathcal{P}$ picks two n-bit strings $\mathbf{x}, \mathbf{y}$ uniformly at random.
  2. $\mathcal{V}$ selects a uniformly random index $i \in [n]$.
  3. $\mathcal{P}$ and $\mathcal{V}$ engage in the PIR protocol PIR, where $\mathcal{P}$ simulates the server on the database $\mathbf{x} = (x_1, \ldots, x_n)$, and $\mathcal{V}$ simulates the user on retrieval index i.
  4. $\mathcal{P}$ sends $\mathbf{y}$ and $c = b \oplus \langle \mathbf{x}, \mathbf{y} \rangle$ to $\mathcal{V}$.

  Reveal phase: To open the commitment, $\mathcal{P}$ sends the string $\mathbf{x}$ to $\mathcal{V}$.
- Ver($PIR, x_i, \mathbf{y}, c, \mathbf{x}$): $\mathcal{V}$ verifies that the string $\mathbf{x}$ is consistent with the bit x_i retrieved by $\mathcal{V}$ during the execution of the protocol PIR in commit phase, and then computes $\langle \mathbf{x}, \mathbf{y} \rangle$ and $b = c \oplus \langle \mathbf{x}, \mathbf{y} \rangle$.

We illustrate the protocol flow in Figure 2.7.

The weak binding property follows from the privacy requirement of the PIR protocol, i.e., the prover $\mathcal{P}$ can open the commitment ambiguously by coming up with a bit string $\mathbf{x}' = (x'_1, \ldots, x'_n) \in \{0,1\}^n$ and an index $j \neq i$ such that $x_i \neq x'_j$ and $c = (1-b) \oplus \langle \mathbf{x}', \mathbf{y} \rangle$. This can be only performed with probability that negligibly greater than $1 - 1/n$. Otherwise the privacy of the underlying PIR protocol is compromised. Breaking the statistically hiding property is reduced to predicting the inner product of two uniformly random bit strings. According to the randomized communication complexity of the inner product shown in [CG88], the advantage of predicting such an inner product is exponentially small, i.e., $1/2 + negl(n)$.

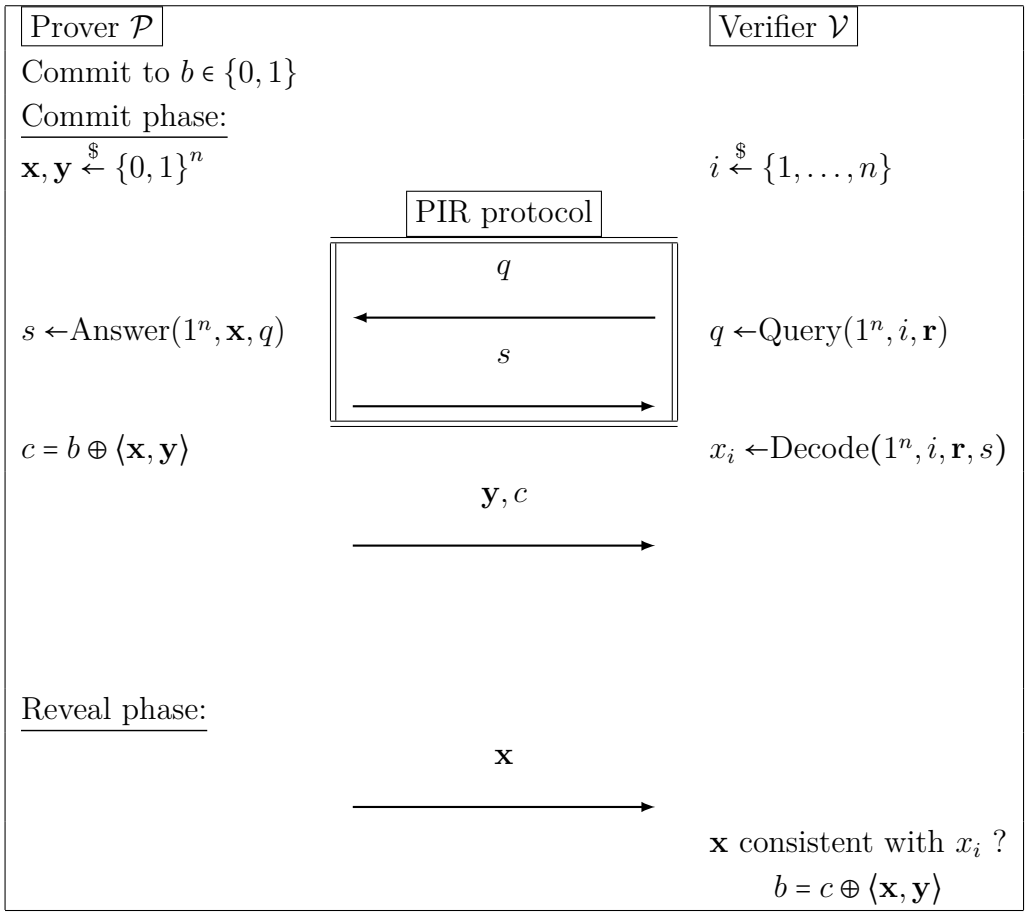

Figure 2.7: Statistically hiding bit commitment protocol from [BIKM99].

Chapter 3

Post-Quantum Commitment Schemes

In this chapter we provide a survey on concrete constructions of post-quantum commitment schemes. The security of their computational (hiding or binding) property is based on computational problems that are believed to be secure even under quantum attacks. Concrete constructions are not general frameworks for commitment schemes like those reviewed in the previous chapter.

As mentioned in the previous chapter, we are only interested in commitment schemes with unconditionally hiding or unconditionally binding property, i.e., we do not consider trapdoor schemes, multi-committer schemes, or universally composable schemes.

In particular, there are many works showing that commitment schemes can be implemented using quantum computing devices, and require quantum computation or exchanging quantum information. The first such scheme is the (flawed) one proposed in [BB84]. Many other better contributions were later proposed like the constructions proposed in [DMS00, Yam13]. However, we focus on *post-quantum* commitment schemes. These are commitment schemes that run on conventional computers, and whose security is believed to hold up against quantum computers.

To the best of our knowledge, there are no other concrete commitment schemes under the stated conditions so far, which are believed to be secure even under quantum attacks.

In Figure 3.1 we give an overview of the commitment schemes we review in this chapter in form of two simple trees. The leafs of both trees include the (sub) sections

where the schemes are reviewed, followed by the respective original papers in which the constructions appeared. The parent (middle) nodes of the leafs include the computational assumptions on which the computational (hiding or binding) property of the commitment schemes are based. These computational assumptions are: small integer solution (SIS) and its ring variant (Ring-SIS), learning parity with noise (LPN), learning with errors (LWE), and the LWE ring variant (RLWE). The two roots represent the resulting schemes, where UHC and UBC stand for unconditionally hiding commitments and unconditionally binding commitments, respectively.

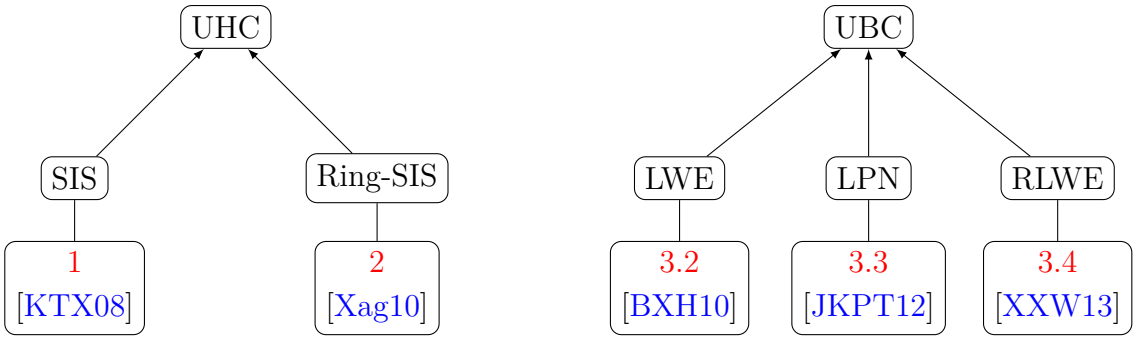

Figure 3.1: Overview of post-quantum commitment schemes.

Section 3.1 reviews the unconditionally hiding commitment schemes proposed in [KTX08, Xag10]. Section 3.2, 3.3, and 3.4 reviews the unconditionally binding commitment schemes proposed in [BXH10, JKPT12, XXW13], respectively.

3.1 Statistically Hiding Commitments from (Ring-) SIS

Hereafter we review two statistically hiding commitment schemes. Both schemes are non-interactive, commit to bit strings, and their binding property is based on the collision resistance of a lattice-based and ideal-lattice-based family of hash functions. More precisely, breaking the binding property implies solving the SIS or Ring-SIS problem, which both are considered as the source of lattice-based and ideal-lattice-based hash functions. Originally, the commitment scheme based on SIS assumption was presented in [KTX08]. Thereafter, an ideal lattice variant of the scheme based on Ring-SIS was shown in [Xag10]. First we recall in Subsection 3.1.1 the definition of the lattice-based and ideal-lattice-based family of hash functions. Then we review in Subsection 3.1.2 the construction of both commitment schemes.

3.1.1 (Ideal-) Lattice-Based Hash Functions

First we recall the definition of the lattice-based family of hash functions. Then we review the definition of the ideal-lattice-based family of hash functions. We basically follow [Xag10] with some details from [MR09, LMPR08].

Definition 3.1 (Lattice-Based Hash Functions)**.** Let n be a security parameter which corresponds to the underlying lattice dimension. For a modulus $q = n^{O(1)}$, a small integer $d \geq 1$, and an integer $m = poly(n) > n \log q / \log d$, let the set

$$\mathcal{F}_n = \{f_{\mathbf{A}} : \{0, \ldots, d-1\}^m \longrightarrow \mathbb{Z}_q^n \mid \mathbf{A} \in \mathbb{Z}_q^{n \times m}\}$$

be a family of hash functions parametrized by q, d, and m, where

- KGen(1^n): Given 1^n, output a uniformly random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$.
- Hash($\mathbf{A}, \mathbf{x}$): Given the key $\mathbf{A}$, and a message $\mathbf{x} \in \{0, \ldots, d-1\}^m$, output the digest

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \pmod q \in \mathbb{Z}_q^n \ .$$

The family of hash functions $\mathcal{F}_n$ defined above was originally presented in [Ajt96]. The work of [Ajt96] shows that $\mathcal{F}_n$ is a family of one-way functions whose security is based on the worst case hardness of GapSVP$_\gamma^2$, for $\gamma = poly(n)$, i.e., being able to invert a function chosen from the family $\mathcal{F}_n$ with non-negligible probability implies the ability to solve any instance of GapSVP$_\gamma^2$.

A followup work in [GGH96] shows that the family is indeed collision resistant for an appropriate choice of q, d, and m. The main statement is that finding a collision $(\mathbf{x_1}, \mathbf{x_2})$ for any $f_{\mathbf{A}} \in \mathcal{F}_n$ means that $\mathbf{A}\mathbf{x_1} \equiv \mathbf{A}\mathbf{x_2} \pmod q$, which immediately yields finding a short non-zero vector $\mathbf{x} = \mathbf{x_1} - \mathbf{x_2}$ such that $\mathbf{A}\mathbf{x} \equiv \mathbf{0} \pmod q$ and $\|\mathbf{x}\|_2 \leq \sqrt{\sum_1^m (d-1)^2} = (d-1)\sqrt{m}$. This means solving SIS$^2_{q,m,(d-1)\sqrt{m}}$.

Many subsequent works were established which all focused on obtaining optimizations and improving the security assumption. Nevertheless, these hash functions are not particularly efficient, because the key size grows at least quadratically in the security parameter n. This is practically considered too large for a simple cryptographic primitive like collision resistant hash function.

A highly optimized and efficient family of hash functions was proposed in [LM06] and [LMPR08] whose collision resistance property is based on the worst case hardness of finding relatively short non-zero vectors in n-dimensional ideal lattices. For

simplicity, we describe below the family of hash functions introduced in [LMPR08], which can be considered as a special case of the family presented in [LM06]. More precisely, the work of [LMPR08] considered a concrete instance from the family presented in [LM06]. This instance enjoys very efficient implementation from a practical point of view.

The security parameter n, which corresponds to the underlying ideal lattice dimension, is a power of two, so that the monic polynomial $x^n + 1$ is irreducible. We define the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, whose elements are residue classes of polynomials of degree less than n with integer coefficients modulo q. For $d = 2$, we define the ring R_d in a similar way.

Definition 3.2 (Ideal-Lattice-Based Hash Functions)**.** Let $\hat{\mathcal{F}}_n$ =(KGen,Hash) be a family of hash functions parametrized by $d = 2$, a prime modulus q such that $2n$ divides $q - 1$, and an integer $m = poly(n) > n \log q$, where

$$\hat{\mathcal{F}}_n = \{f_{a_1,\ldots,a_{m/n}} : R_d^{m/n} \longrightarrow R_q \mid a_1, \ldots, a_{m/n} \in R_q\} .$$

The algorithms KGen and Hash are defined as follows.

- KGen(1^n): Given 1^n, output random fixed polynomials $a_1, \ldots, a_{m/n} \xleftarrow{\$} R_q$.
- Hash($(a_1, \ldots, a_{m/n}), \mathbf{x}$): Given the key $(a_1, \ldots, a_{m/n})$, and a message $\mathbf{x} = (x_1, \ldots, x_{m/n}) \in R_d^{m/n}$, output the digest

$$f_{a_1,\ldots,a_{m/n}}(\mathbf{x}) = \sum_{i=1}^{m/n} a_i \cdot x_i \pmod{q} \in R_q .$$

For $d = 2$, the original representation from [Ajt96] has a key size of at most $nm\lceil \log q \rceil$ bits, while the construction from [LMPR08] has a key size of at most $m\lceil \log q \rceil$ bits. Moreover, the main technique underlying the construction from [LMPR08] is the use of *Fast Fourier Transform* (FFT) over the ring R_q in order to perform the operations $a_i \cdot x_i$ very efficiently. Finally, finding a collision for any $f_{a_1,\ldots,a_{m/n}} \in \hat{\mathcal{F}}_n$ on the average with any non-negligible probability means solving Ring-SIS$^{\infty}_{q,m/n,\beta}$ over the ring R_q, where β is the upper bound of the ℓ_∞ norm of vectors in $R_d^{m/n}$.

Remark 3.1. Unlike the special family defined in [LMPR08], the ideal lattice family of hash functions presented in [LM06] can be instantiated using any monic and irreducible polynomial g of degree $n \in \mathbb{N}$ from the polynomial ring $\mathbb{Z}[x]$, i.e., g is not restricted to be the polynomial $x^n + 1$, where n is a power of two. Furthermore, the modulus q is not necessarily prime, and the parameter d is not necessarily 2.

3.1.2 Commitments from (Ring-) SIS

In the following we review the two non-interactive statistically hiding commitment schemes presented in [KTX08, Xag10] which use the lattice and ideal lattice family of hash functions given in Definition 3.1 and 3.2, respectively.

I. Lattice-based commitment scheme

The first commitment scheme, shown in [KTX08], uses the lattice-based family of hash functions $\mathcal{F}_n$ given in Definition 3.1. The scheme is a direct application of the family $\mathcal{F}_n$ with the parameters $q \geq 4mn^{3/2}$, $m > 10n \log q$, and $d = 2$, i.e., the message space of the family $\mathcal{F}_n$ are m-bit strings. The matrix $\mathbf{A}$ is wide enough such that it can be multiplied by the concatenation of the message $\mathbf{m}$ being committed to, and a random bit string $\mathbf{r}$.

- Setup(1^n): The setup algorithm samples a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and sets $\mathbf{A}$ as a public commitment key.
- Com$(\mathbf{A}, \mathbf{m})$: The algorithm Com works as follows.
  Commit phase: To commit to a message $\mathbf{m} \in \{0,1\}^{m/2}$, the prover $\mathcal{P}$ samples a uniformly random bit string $\mathbf{r} \in \{0,1\}^{m/2}$, computes
  $$\mathbf{c} = \mathbf{A}(\mathbf{r} \| \mathbf{m}) \pmod q \in \mathbb{Z}_q^n,$$
  and sends the commitment $\mathbf{c}$ to the verifier $\mathcal{V}$.
  Reveal phase: To open the commitment, $\mathcal{P}$ sends the message $\mathbf{m}$, and the randomness $\mathbf{r}$ to $\mathcal{V}$.
- Ver$(\mathbf{A}, \mathbf{c}, \mathbf{m}, \mathbf{r})$: The verifier $\mathcal{V}$ checks that $\mathbf{m}, \mathbf{r} \in \{0,1\}^{m/2}$ and verifies that $\mathbf{A}(\mathbf{r} \| \mathbf{m}) = \mathbf{c} \pmod q$.

The computationally binding property simply follows from the collision resistance property of the family $\mathcal{F}_n$. On the other hand, the statistically hiding property is derived from a claim shown in [Reg09] which informally states that for some integer ℓ and ℓ uniformly random elements $g_1, \ldots, g_\ell$ from some finite Abelian group, a random subset sum $\sum_{i \in [\ell]} r_i g_i$ for uniformly random elements $r_i \in \{0,1\}$ is statistically close to the uniform distribution for almost all choices of the elements $g_1, \ldots, g_\ell$. In the above commitment scheme, the finite Abelian group is the set $\mathbb{Z}_q^n$, and the columns of the matrix $\mathbf{A}$, denoted by $\mathbf{a}_i \in \mathbb{Z}_q^n$ for all $i \in [m]$, are the group elements

chosen uniformly at random. Hence, the commitment $\mathbf{c}$ can be written as

$$\mathbf{c} = \mathbf{A}(\mathbf{r}\|\mathbf{m}) = \sum_{i=1}^{m/2} r_i\mathbf{a}_i + \sum_{i=1}^{m/2} m_i\mathbf{a}_{i+m/2} \pmod q,$$

where r_i, m_i are the entries of $\mathbf{r}, \mathbf{m}$, respectively. Following the claim implies that the subset sum $\sum_{i=1}^{m/2} r_i\mathbf{a}_i$ is statistically close to the uniform distribution with overwhelming probability over the choice of the columns $\mathbf{a}_i \in \mathbb{Z}_q^n$. This shows that for all but negligible fraction of choice of the matrix $\mathbf{A}$, the distributions of two commitments are statistically close.

The following theorem, shown in [KTX08], summarizes the previous results.

Theorem 3.1. *Let $q \geq 4mn^{3/2}$, $m > 10n\log q$, and $d = 2$. If $SIS^2_{q,m,\sqrt{m}}$ is hard on the average, then Protocol 1 is a statistically hiding and computationally binding string commitment scheme. In particular, if $GapSVP^2_\gamma$ is hard in the worst case, for $\gamma = 14\pi\sqrt{nm}$, then Protocol 1 is a statistically hiding and computationally binding string commitment scheme.*

As suggested in [KTX08], the message space can be simply extended using the Merkle-Damgård technique [Mer90, Dam90], which uses an appropriate padding function and a collision resistant compression function for a fixed message length to obtain a collision resistant hash function for messages with arbitrary length. Applying the Merkle-Damgård construction to the above commitment function

$$\{0,1\}^{m/2} \times \{0,1\}^{m/2} \longrightarrow \mathbb{Z}_q^n, \quad (\mathbf{m},\mathbf{r}) \longrightarrow \mathbf{A}(\mathbf{r}\|\mathbf{m}) \pmod q,$$

yields the function

$$\{0,1\}^* \times \{0,1\}^{m/2} \longrightarrow \mathbb{Z}_q^n, \quad (\mathbf{m},\mathbf{r}) \longrightarrow \mathbf{A}(\mathbf{r}\|\mathbf{m}) \pmod q .$$

II. Ideal-lattice-based commitment scheme

Next we use the ideal-lattice-based family of hash functions $\hat{\mathcal{F}}_n$, defined in [LMPR08] and given in Definition 3.2, to review an instance of the ring variant of the commitment scheme presented in [KTX08]. More precisely, the scheme we describe below is an instance of the Ring-SIS-based construction presented in [Xag10]. This instance uses the ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$, where q is prime, and n is a power of two. To commit to a bit string, the construction given in [Xag10] selects a random string, whose entries are from the set $\{-1, 0, +1\}$. The instance we describe below restricts this set to $\{0,1\}$ as in the SIS-based commitment scheme given in [KTX08] [1]. This

[1] This restriction does not affect the security of the commitment scheme, since the resulting scheme corresponds to the one given in [KTX08].

restriction simplifies the comparison that we later perform between the SIS-based and the Ring-SIS-based commitment scheme (see Subsection 4.2.1).

- Setup(1^n): The setup algorithm samples uniformly random fixed polynomials $a_1, \ldots, a_{m/n} \xleftarrow{\$} R_q$, and sets $(a_1, \ldots, a_{m/n})$ as a public commitment key.
- Com($(a_1, \ldots, a_{m/n}), \mathbf{m}$): The algorithm Com works as follows.
  Commit phase: To commit to a message $\mathbf{m} \in \{0,1\}^{m/2}$, the prover $\mathcal{P}$ samples a uniformly random bit string $\mathbf{r} \in \{0,1\}^{m/2}$, computes the hash value
  $$c = f_{a_1,\ldots,a_{m/n}}(\mathbf{r} \| \mathbf{m}),$$
  and sends the commitment $c \in R_q$ to the verifier $\mathcal{V}$.
  Reveal phase: To open the commitment, $\mathcal{P}$ sends the message $\mathbf{m}$, and the randomness $\mathbf{r}$ to $\mathcal{V}$.
- Ver($(a_1, \ldots, a_{m/n}), c, \mathbf{m}, \mathbf{r}$): $\mathcal{V}$ verifies that $\mathbf{m}, \mathbf{r} \in \{0,1\}^{m/2}$, computes the hash value of $\mathbf{r} \| \mathbf{m}$, and compares the result with c.

The following theorem, shown in [Xag10], establishes the security of the commitment scheme described above.

Theorem 3.2. *Let $E = EF(x^n + 1, 3) \leq 3$ [2], $\Delta = \frac{1}{2}\Delta(q, x^n + 1, 3)$, $q > 3Em\sqrt{n}\log n$, $m > 4n \log q$, and $d = 2$. If Ring-SIS$^{\infty}_{q,m/n,1}$ is hard on the average, and the statistical distance Δ is negligible in n, then Protocol 2 is a statistically hiding and computationally binding string commitment scheme. In particular, if SVP$^{\infty}_{\gamma}$ is hard in the worst case for $\gamma = 8E^2 m \log^2 n$, and Δ is negligible in n, then Protocol 2 is a statistically hiding and computationally binding string commitment scheme.*

According to [Xag10], the message space of the ideal-lattice-based commitment scheme described above can also be extended using the Merkle-Damgård technique to obtain a commitment function for bit strings with arbitrary length.

[2] The value $EF(x^n + 1, 3)$ is the expansion factor of the polynomial $x^n + 1$ (see Equation 1.3).

3.2 Statistically Binding Commitments from LWE-Based Encryption

This section describes the commitment scheme proposed in [BXH10], which employs the LWE-based public-key cryptosystem presented in [GPV08].

First we review the required parameters for the cryptosystem, proposed in [GPV08], in order to decrypt correctly with overwhelming probability, and achieve the required security assuming the hardness of LWE. For a security parameter n, the cryptosystem uses the parameters $r \geq \omega(\sqrt{\log m})$, $\alpha \leq 1/(r\sqrt{m} \cdot \omega(\sqrt{\log n}))$, $m \geq 2(n+1)\ln q$, and a prime modulus $q \geq 5rm$. Furthermore, the cryptosystem sets $\bar{\Psi}_\alpha$ to be the distribution on $\mathbb{Z}_q$ obtained by sampling a normal variable with mean 0 and standard deviation $\alpha q/\sqrt{2\pi}$, rounding the result to the nearest integer and reducing it modulo q.

Next we recall the LWE-based public-key cryptosystem from [GPV08].

- KGen(1^n): Given 1^n, choose a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$, a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and compute the vector $\mathbf{p} = \mathbf{A}^T\mathbf{s} + \mathbf{x} \in \mathbb{Z}_q^m$, where each entry x_i from $\mathbf{x}$ is chosen independently from the error distribution $\bar{\Psi}_\alpha$ for all $i \in [m]$. Finally, set $sk = \mathbf{s}$, $pk = (\mathbf{A}, \mathbf{p})$ as the private and public key, respectively.
- Enc($(\mathbf{A}, \mathbf{p}), b$): Given a public key $pk = (\mathbf{A}, \mathbf{p})$, and a bit $b \in \{0, 1\}$ to encrypt, choose a vector $\mathbf{r} \in \mathbb{Z}^m$ from the discrete Gaussian distribution $D_{\mathbb{Z}^m, r}$, and compute $\mathbf{u} = \mathbf{A}\mathbf{r}, c = \mathbf{p}^T\mathbf{r} + b \cdot \lfloor q/2 \rfloor$. Finally, output the ciphertext $(\mathbf{u}, c) \in \mathbb{Z}_q^{n+1}$.
- Dec($\mathbf{s}, (\mathbf{u}, c)$): Given a secret key $sk = \mathbf{s}$, and a ciphertext $(\mathbf{u}, c)$, compute $b' = c - \mathbf{s}^T\mathbf{u} \in \mathbb{Z}_q$, and output the bit 0 if b' is closer to 0 than to $\lfloor q/2 \rfloor$ modulo q, otherwise output 1.

The following lemma, shown in [GPV08], indicates that it is possible to extract the secret key $sk = \mathbf{s}$ from a properly-generated public key $pk = (\mathbf{A}, \mathbf{p})$ using a trapdoor $\mathbf{T}$ for the matrix $\mathbf{A}$.

Lemma 3.1. *There exists an efficient algorithm for the above cryptosystem such that for all but at most negligible fraction of public keys generated by the algorithm KGen(1^n), given a trapdoor $\mathbf{T}$ for the matrix $\mathbf{A}$ and a public key $(\mathbf{A}, \mathbf{p})$, it can efficiently extract the unique secret key $\mathbf{s}$.*

We proceed to review the statistically binding commitment scheme proposed in [BXH10], which is non-interactive, and commits to single bits. The scheme directly employs the public-key cryptosystem described above. We denote this public-key cryptosystem by $\mathcal{E}$.

- Setup(1^n): The setup algorithm chooses some security parameter $n \in \mathbb{N}$ for the public-key cryptosystem $\mathcal{E}$ described above.
- Com(n, b): The algorithm Com works as follows.
  Commit phase: To commit to a bit $b \in \{0, 1\}$, the prover $\mathcal{P}$ chooses parameters $r, m, q, \bar{\Psi}_\alpha$ for the cryptosystem $\mathcal{E}$, and invokes the algorithm KGen and Enc of $\mathcal{E}$ to obtain a public key and a ciphertext, i.e.,

  $$(pk = (\mathbf{A}, \mathbf{p}), sk = \mathbf{s}) \leftarrow \text{KGen}(1^n), \quad \text{and} \quad (\mathbf{u}, c) \leftarrow \text{Enc}((\mathbf{A}, \mathbf{p}), b),$$

  where the matrix $\mathbf{A}$ admits a trapdoor $\mathbf{T}$. The commitment $\mathbf{c}$ consists of the chosen parameters, and the public key along with the encryption pair, i.e.,

  $$\mathbf{c} = (r, m, q, \bar{\Psi}_\alpha, \mathbf{A}, \mathbf{p}, \mathbf{u}, c) \ .$$

  Reveal phase: To open the commitment, $\mathcal{P}$ sends to the verifier the committed bit b, the trapdoor $\mathbf{T}$ of the matrix $\mathbf{A}$, and the vectors $\mathbf{x}, \mathbf{r}$ used in generating the public key and the encryption pair, respectively.
- Ver$(n, \mathbf{c}, b, \mathbf{T}, \mathbf{x}, \mathbf{r})$: The verifier $\mathcal{V}$ checks that the commitment is legally generated by computing the public key and the ciphertext. Moreover, $\mathcal{V}$ verifies that the secret key is extractable and corresponds to the public key.

The extraction algorithm guaranteed by Lemma 3.1 is deterministic. This implies that almost all legitimate public keys correspond to a unique private key. Thus, the bit committed by an encryption relative to such legitimate public key has a unique decryption. Therefore, it holds the statistically binding property. Furthermore, the computationally hiding property follows directly from the security of the cryptosystem.

Remark 3.2. An extension of the above commitment scheme for committing to ℓ-bit strings was appeared in [BLX11]. More precisely, the length ℓ is set to be public along with the security parameter n. The authors stated that their scheme is perfectly binding. The commitment of an ℓ-bit string $(b_1, b_2, \ldots, b_\ell)$ is the tuple

$$\mathbf{c} = \Big(r, m, q, \bar{\Psi}_\alpha, \mathbf{A}, \mathbf{p}, \text{Enc}((\mathbf{A}, \mathbf{p}), b_1), \text{Enc}((\mathbf{A}, \mathbf{p}), b_2), \ldots, \text{Enc}((\mathbf{A}, \mathbf{p}), b_\ell)\Big) \ .$$

3.3 Perfectly Binding Commitments from LPN

We review below a non-interactive perfectly binding commitment scheme proposed in [JKPT12]. The scheme commits to bit strings, and its hiding property is based on the *exact learning parity with noise* (xLPN) assumption. The assumption was defined in [JKPT12], which is a new version of the LPN problem. The $\text{xLPN}_{n,\tau}$ problem is defined exactly like $\text{LPN}_{n,\tau}$, except that the Hamming weight ω of the error vector is exactly $\lfloor t\tau \rceil$, where t is the length of the error vector, and $0 < \tau < 1/2$ is the noise parameter, i.e., the error vector is sampled uniformly at random from the set of all t-bit strings of Hamming weight $\omega = \lfloor t\tau \rceil$. We denote this set by $\{0,1\}^t_\omega$.

The commitment scheme is parameterized by a security parameter n, which is chosen such that $\text{LPN}_{n,\tau}$ is hard, where $0 < \tau < 1/4$. Furthermore, the scheme uses the parameters $t \in O(n+\ell), \omega = \lfloor t\tau \rceil$, where ℓ is the length of the message being committed to. The message length ℓ can be arbitrary. However, to perform efficiency, it was suggested in [JKPT12] to choose it roughly of the same size as n.

- Setup(1^n): The algorithm sets a binary matrix $\mathbf{A} = \mathbf{A}' \| \mathbf{A}'' \in \{0,1\}^{t\times(n+\ell)}$ as a public commitment key, where $\mathbf{A}' \in \{0,1\}^{t\times n}$, and $\mathbf{A}'' \in \{0,1\}^{t\times\ell}$ are chosen uniformly at random.
- Com$(\mathbf{A}, \mathbf{m})$: The algorithm Com works as follows.
  Commit phase: To commit to a message $\mathbf{m} \in \{0,1\}^\ell$, the prover $\mathcal{P}$ selects uniformly random strings $\mathbf{r} \in \{0,1\}^n, \mathbf{e} \in \{0,1\}^t_\omega$, compute
  $$\mathbf{c} = \mathbf{A}(\mathbf{r}\|\mathbf{m}) \oplus \mathbf{e} = \mathbf{A}'\mathbf{r} \oplus \mathbf{A}''\mathbf{m} \oplus \mathbf{e},$$
  and sends the commitment $\mathbf{c} \in \{0,1\}^t$ to the verifier $\mathcal{V}$.
  Reveal phase: To open the commitment, $\mathcal{P}$ sends $\mathbf{m}$ and $\mathbf{r}$ to $\mathcal{V}$.
- Ver$(\mathbf{A}, \mathbf{c}, \mathbf{m}, \mathbf{r})$: $\mathcal{V}$ verifies that the bit string $\mathbf{c} \oplus \mathbf{A}(\mathbf{r}\|\mathbf{m})$ has Hamming weight exactly ω.

Setting $t \in \Theta(n+\ell)$ large enough implies that the distance of the code generated by the random matrix $\mathbf{A} \in \{0,1\}^{t\times(n+\ell)}$ is large. Thus, it follows the perfectly binding property. Moreover, the computationally hiding property follows directly from the $\text{xLPN}_{n,\tau}$ assumption, which implies that $\mathbf{A}'\mathbf{r} \oplus \mathbf{e}$, and hence $\mathbf{c}$, is pseudorandom.

3.4 Perfectly Binding Commitments from RLWE

Eventually, we review a perfectly binding commitment scheme presented in [XXW13]. The scheme is non-interactive, commits to vectors of integers from a finite ring, and its hiding property is based on the RLWE assumption. This work can be seen as an extended construction of the scheme from the previous section, which was shown in [JKPT12], since LPN can be considered as a special case of the LWE problem.

The security parameter n is a power of two, and the message space is R_q^ℓ for $\ell = poly(n)$, where $R = \mathbb{Z}[x]/(\Phi_m(x))$, $\Phi_m(x)$ is the m-th cyclotomic polynomial of degree $\varphi(m) = n$, and $R_q = R/qR$ for a prime $q = poly(n)$ such that $q = 1 \mod m$. Additionally, the scheme uses the parameters $t = (\ell + 1) \cdot \omega(\log n)$, $\beta < q/2n$, and a β-bounded distribution χ over R, i.e., χ outputs elements of R with norm at most β with overwhelming probability. To perform efficiency, it was suggested in [XXW13] to choose $\ell \in O(1)$, and $t = \omega(\log n)$.

- Setup(1^n): The setup algorithm samples $\mathbf{a} \xleftarrow{\$} R_q^t$, $\mathbf{A}' \xleftarrow{\$} R_q^{t\times\ell}$, and sets $\mathbf{A} = \mathbf{a}\|\mathbf{A}' \in R_q^{t\times(\ell+1)}$ as a public commitment key.
- Com$(\mathbf{A}, \mathbf{m})$: The algorithm Com works as follows.
  Commit phase: The commitment of a message $\mathbf{m} \in R_q^\ell$ is the value
  $$\mathbf{c} = \mathbf{A}(r\|\mathbf{m}) + \mathbf{e} = \mathbf{a}r + \mathbf{A}'\mathbf{m} + \mathbf{e} \in R_q^t,$$
  where $r \in R_q$ is chosen uniformly at random, and $\mathbf{e} \leftarrow \chi^t$.
  Reveal phase: Opening the commitment is performed by sending the message $\mathbf{m}$, and the randomness r.
- Ver$(\mathbf{A}, \mathbf{c}, \mathbf{m}, r)$: The verifier checks if $\|\mathbf{c} - \mathbf{A}(r\|\mathbf{m})\|_\infty \leq \beta$.

The perfectly binding property is due to the fact that the shortest non-zero vector in the q-ary lattice defined by the matrix $\mathbf{A}$ is not too small with overwhelming probability. The computationally hiding property follows directly from the $\mathrm{RLWE}_{q,\chi}$ assumption, i.e., $\mathbf{c}$ is pseudorandom.

Remark 3.3. The commitment scheme described above was proposed in [BKLP14] as well, but with a slight difference such that requirements on valid openings are relaxed to be able to realize better zero-knowledge proofs while still preserving the perfectly binding property of the scheme. More precisely, the ring is set to $R = \mathbb{Z}[x]/(x^n + 1)$, the parameter ℓ is chosen to be 1, i.e., the message space is R_q, and an additional small polynomial $f \in R_q$ is allowed in valid openings, where

$\|f\|_\infty \leq 1$, and $\deg(f) < n/2$. An honest prover can always set $f = 1$ when opening the commitment $\mathbf{c}$. Therefore, the verification is not affected by this relaxation, while a malicious prover still cannot open a commitment to two different messages.

Chapter 4

Comparing Commitment Schemes

In this chapter we compare the commitment schemes included in Chapter 2 and 3. The goal of the comparison is to evaluate the properties that each scheme enjoys, and to obtain two schemes, one that is unconditionally hiding, and one that is unconditionally binding. Both commitment schemes are desired to be "efficient" and secure even against quantum computers.

In fact, the efficiency of a protocol depends on different factors. The most important factors are *interaction* and *communication complexity*. Protocols are typically interactive, since their parties exchange messages back and forth. Thus, interaction is very expensive, because the number of rounds of communication heavily reflects on the overall running time of a protocol. Communication complexity refers to the total number of bits sent by the parties during the execution of a protocol and plays a major role for its efficiency. In our comparison we consider these two factors to measure the efficiency of the commitment schemes.

We separately compare unconditionally hiding commitment schemes, and unconditionally binding commitment schemes among each other. This is reasonable and meaningful, since unconditionally hiding and binding commitments have different applications. For example, unconditionally hiding commitments are required when commitment schemes are used as a building block in constructions of larger protocols. In such applications, the binding property is typically required to ensure the unambiguity of commitments that are opened during the protocol execution. However, the hiding property is required to ensure that the unopened commitments remain secret even after the protocol execution. Thus, it is sufficient for the binding property to be computational, whereas the hiding property is required to hold unconditionally. On the other hand, commitment schemes were used in the work of

[GMW91] on building zero-knowledge interactive proof systems for every language in NP. In this work, it is sufficient for the hiding property to be computational, whereas the binding property is required to hold unconditionally.

We start in Section 4.1 by comparing the general constructions of commitment schemes from Chapter 2. Then we compare the concrete commitment schemes from Chapter 3 in Section 4.2. Finally, we compare in Section 4.3 the results of Section 4.1 with the results of Section 4.2.

We give an overview of our comparison plan in Figure 4.1, where UHC and UBC stand for unconditionally hiding and unconditionally binding commitment schemes, respectively.

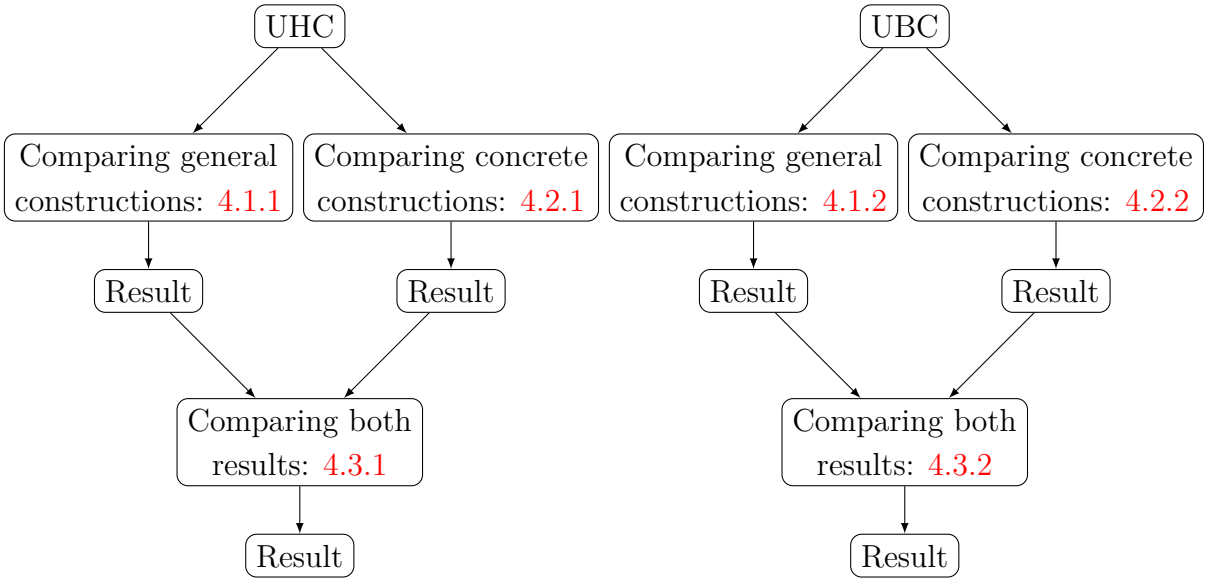

Figure 4.1: Overview of comparing the commitment schemes.

4.1 Comparison Between General Constructions

In this section we compare the general constructions of commitment schemes from Chapter 2. We start by comparing the constructions of unconditionally hiding commitment schemes with each other. Then, we compare the constructions of unconditionally binding commitment schemes with each other.

4.1.1 Unconditionally Hiding

We first provide a brief review of the general constructions that provide unconditionally hiding commitment schemes in form of a table (see Table 4.1). This table includes some features, which are set in columns. We declare these features in the following.

In the first column we refer to the corresponding subsection and original paper of each protocol. The second column includes the model of each protocol, which indicates if the scheme is statistically (sta.) or perfectly (per.) hiding. We then consider the protocol interaction in the third column, i.e., we provide the number of rounds (or an upper bound) required for sending the commitment. The next two columns include the message type being committed to (bit or bit string), and the computational assumption or protocol, on which each construction is based. This includes one-way functions (OWF), one-way permutations (OWP), collision resistant hash functions (CRHF), Σ-protocols, and PIR protocols. The last column is for communication overhead, i.e., we give in this column the total number of bits (or an upper bound) required for sending and opening a commitment. For the rest of this chapter, the security parameter of all schemes is denoted by n.

The scheme	Model	Interaction	Message type	Computational assumption	Communication complexity
2.1.2 [HNO+09]	sta.	$n^{O(1)}$ rounds	bit	OWF	$poly(n)$
2.2.2 [NOVY98]	per.	$2n-1$ rounds	bit	OWP	n^2+n+1
2.2.3 [TPY08]	per.	2 rounds	bit	OWP	$3n$
2.3.1 [HM96]	sta.	1 round	bit string	CRHF	$\ell + O(n)^\star$
2.3.2 [DPP97]	sta.	≥ 2 rounds	bit/bit string	CRHF	$poly(n)$
2.3.3 [DPP98]	sta.	1 round	bit string	CRHF	$\ell + O(n)^\star$
2.5.1 [Dam89]	per.	1 round	bit string	Σ-protocol	$poly(n)$
2.5.2 [BIKM99]	sta.	$\geq 3n^2$ rounds	bit	PIR protocol	$poly(n)$

* ℓ is the message length.

Table 4.1: A brief review of the unconditionally hiding schemes from Chapter 2.

Assuming that the hiding property is not required to hold perfectly, Table 4.1 shows that there are only two constructions that are interesting for the comparison. Namely, the construction [HM96] and [DPP98]. In the following we justify this statement by gradually excluding the other constructions from consideration and giving a significant reason for the exclusion.

- The authors of [HNO+09] stated that their construction is very inefficient and

certainly would never be utilized in practice. This can also be seen from Table 4.1, since the construction utilizes $n^{O(1)}$ rounds of interaction and $poly(n)$ bits of communication for committing to only a single bit. The same reasoning applies to the construction [BIKM99] which employs some PIR protocol.

- By comparing the constructions [NOVY98, TPY08] which both use OWPs, we observe that the construction [NOVY98] is also inefficient. This is because it calls for $2n-1$ rounds to commit to a single bit and requires polynomially many bits of communication. On the other hand, the construction [TPY08] calls for only two rounds and requires linearly many bits of communication. Hence, it is obviously more efficient than [NOVY98]. Nevertheless, there is unfortunately no one-way permutation known so far, which is believed to be secure under quantum attacks.

- Although the construction [DPP97] may have communication cost of $O(n)$ rather than $poly(n)$ bits, we observe that it requires at least two rounds. This depends on the key generation algorithm of the utilized fail-stop signature scheme which can be constructed from any CRHF. In contrast, the constructions [HM96, DPP98] are non-interactive and use CRHFs as well. Therefore, provided that the construction [DPP97] achieves $O(n)$ bits of communication, and it can be used for committing to bit strings, the drawback of this construction is then the interaction, compared with the constructions [HM96, DPP98].

- The construction [Dam89] may have communication cost of $O(n)$ rather than $poly(n)$ bits. This depends on the complexity of the utilized Σ-protocol. However, the utilized Σ-protocol must have the perfect zero-knowledge property. In addition, the hard relation on which the utilized Σ-protocol is defined, must have the property that it is easy to verify membership in the language defined by this relation. Nevertheless, the construction is suitable for applications, where the hiding property is required to hold perfectly.

As a first result, the construction [HM96] and [DPP98] are left to compare. In fact, they are very similar to each other. Both constructions are non-interactive, statistically hiding, commit to ℓ-bit strings, use some family of 2-universal hash functions, and require $\ell + O(n)$ bits of communication.

In order to get precise values for the communication complexity and obtain concrete results, we assume that both commitment schemes have the same security parameter n, and use the same CRHF f. Thus, we first have to determine which family of 2-universal hash functions is better to use. We have seen two constructions of such a family in Example 2.1 and 2.2. Any function from Construction I given in Example

2.1 can be described using $t + 2\ell - 1$ bits, where t, ℓ are the bit length of the input and output, respectively. On the other hand, any function from Construction II given in Example 2.2 requires $2t$ bits, where t is the bit length of the input as well. Therefore, determining which construction is more efficient, depends on the value of t and ℓ. We distinguish the following cases.

- Case 1: If $t > 2\ell - 1$, then Construction I is more efficient than II.
- Case 2: If $t = 2\ell - 1$, then Construction I and II are equivalent.
- Case 3: If $t < 2\ell - 1$, then Construction II is more efficient than I.

The parameters t, ℓ have different values in the commitment schemes [HM96, DPP98]. The former scheme sets $t = 6n + 4$, and $\ell = n$, whereas the latter sets $t = 3(n + 1)$, and $\ell = n + 1$. Thus, for both schemes, the parameter t is greater than $2\ell - 1$, which corresponds to Case 1 given above. Therefore, Construction I given in Example 2.1 is better to use in either commitment scheme. Hence, we employ Construction I for our comparison as a family of 2-universal hash functions $\mathcal{H} = (\mathrm{KGen}(1^n), H)$. Any function $H_k \in \mathcal{H}$ requires $t + 2\ell - 1 = 8n + 3$ bits for the commitment scheme [HM96], and $5n + 4$ bits for the scheme [DPP98].

Table 4.2 gives the required details for the comparison.

The scheme	[HM96]	[DPP98]
Message	$\mathbf{m} \in \{0,1\}^{\ell}$	$\mathbf{m} \in \{0,1\}^{\ell}$
Parameter t	$t = 6n + 4$	$t = 3(n + 1)$
CRHF f	$f : \{0,1\}^{t} \longrightarrow \{0,1\}^{n}$	$f : \{0,1\}^{*} \longrightarrow \{0,1\}^{n+1}$
$\mathcal{H} = (\mathrm{KGen}(1^n), H)$	$H_k : \{0,1\}^{t} \longrightarrow \{0,1\}^{n}$	$H_k : \{0,1\}^{t} \longrightarrow \{0,1\}^{n+1}$
Random string	$\mathbf{r} \in \{0,1\}^{t}$	$\mathbf{r} \in \{0,1\}^{t}$
Commitment string	$(H_k, f(\mathbf{r})) \in \{0,1\}^{9n+3}$	$f\Big(H_k \Vert f(\mathbf{r}) \Vert f(\mathbf{m}) \oplus H_k(\mathbf{r})\Big) \in \{0,1\}^{n+1}$
Decommitment string	$(\mathbf{r}, \mathbf{m}) \in \{0,1\}^{6n+4+\ell}$	$(H_k, \mathbf{r}, \mathbf{m}) \in \{0,1\}^{8n+7+\ell}$
Communication complexity	$15n + 7 + \ell$	$9n + 8 + \ell$

Table 4.2: Comparison between the construction [HM96] and [DPP98].

Finally, Table 4.2 shows that the construction [DPP98] requires lower communication complexity than the construction [HM96]. Therefore, the construction [DPP98] is the result of comparing the general constructions of unconditionally hiding commitment schemes from Table 4.1.

Remark 4.1. We note that our comparison assumed that the hiding property is not required to hold perfectly. If a post-quantum commitment scheme with the

perfectly hiding property is required, then the construction [Dam89] is the only one available from the general constructions of commitment schemes.

4.1.2 Unconditionally Binding

As in the previous subsection, we first give in Table 4.3 a brief review of the general constructions from Chapter 2, which provide unconditionally binding commitment schemes.

The scheme	Model	Interaction	Message type	Computational assumption	Communication complexity
2.1.1 [Nao91]	sta.	2 rounds	bit	OWF	$7n+1$
2.1.1 [Nao91]	sta.	2 rounds	bit string	OWF	$O(\ell), \ell \geq O(n)^{\star}$
2.2.1 [GK96]	per.	1 round	bit	OWP	$3n+1$
2.4.1 [CD04]	sta.	1 round	bit string	PKE**	$poly(n)$

* ℓ is the message length.

** PKE means public-key encryption.

Table 4.3: A brief review of the unconditionally binding schemes from Chapter 2.

Table 4.3 shows that the construction [GK96] is the most efficient one, since it is non-interactive, and requires only $3n+1$ bits of communication. The drawback is that it only commits to single bits. Nevertheless, we unfortunately cannot consider this construction, since there is no one-way permutation known so far, which is believed to be secure under quantum attacks.

The construction [CD04] is non-interactive and performs an additional property beside hiding and binding. Namely, the scheme is *non-repudiable*, i.e. a prover cannot deny having committed to any bit string, that already has been committed to by the prover. Non-repudiability can be achieved in any commitment scheme by having the commitment value digitally signed by the prover using some digital signature scheme before sending the commitment to a verifier. The construction [CD04] utilizes the signing process implicitly so that there is no need for additionally signing commitment values to achieve non-repudiability. However, this construction uses public-key cryptosystems with commutative keys. According to [KL07], the corresponding signature scheme of such a cryptosystem is in most cases inapplicable, and in cases when it is applicable, it results in signature schemes that are completely insecure. Thus, there may be no such a post-quantum cryptosystem. Moreover, public-key cryptography is in general more expensive than symmetric cryptography

such as one-way functions (OWFs) or pseudorandom generators (PRGs). Therefore, the construction [Nao91] that commits to bit strings is less expensive than [CD04], since it uses some PRG.

On the other hand, committing to single bits is much limited, whereas committing to bit strings is a fundamental need in many cryptographic applications. Therefore, it is reasonable to only consider the construction [Nao91] that commits to bit strings, and adopt it as our result of comparing the constructions from Table 4.3.

Remark 4.2. We note that the general constructions of unconditionally binding commitment schemes given in Table 4.3 cannot provide a post-quantum commitment scheme with the perfectly binding property.

4.2 Comparison Between Concrete Constructions

In this section we compare the commitment schemes from Chapter 3. We start by comparing the unconditionally hiding commitment schemes with each other. Then, we compare the unconditionally binding commitment schemes with each other. In addition to the features included in the tables of the previous section, we consider in this section homomorphic properties of the schemes. This is significant, since commitment schemes can be even more useful if they are homomorphic.

4.2.1 Unconditionally Hiding

As in the previous section, we first give in Table 4.4 an overview of the unconditionally hiding commitment schemes from Chapter 3.

By considering our criteria for measuring the efficiency of commitment schemes, Table 4.4 shows that both commitment schemes are non-interactive, and have communication cost of $m + n\lceil \log q \rceil$ bits. The modulus q, and the parameter m differ in either scheme. In order to simplify the comparison, we set the security parameter to $n = 128$, and obtain bounds on q and m given in Table 4.5.

Table 4.5 indicates that the parameters q, m have smaller bounds in the scheme [Xag10] than those in the scheme [KTX08]. Thus, we conclude that the scheme [Xag10] has lower communication complexity than that of the scheme [KTX08].

Remark 4.3. We stress that our comparison considered the interaction and communication complexity, which we set in the beginning of this chapter. However, if

The scheme	1 [KTX08]	2 [Xag10]
Model	statistically	statistically
Interaction	one round	one round
Type of the message $\mathbf{m}$	$(m/2)$-bit string	$(m/2)$-bit string
Computational assumption	SIS	Ring-SIS
Modulus q	$q \geq 4mn^{3/2}$	$q > 3Em\sqrt{n}\log n,\ E \leq 3$
Parameter m	$m > 10n\log q$	$m > 4n\log q$
Public key	$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$	$\mathbf{a} \xleftarrow{\$} R_q^{m/n}$
Random string	$\mathbf{r} \in \{0,1\}^{m/2}$	$\mathbf{r} \in \{0,1\}^{m/2}$
Commitment string	$\mathbf{A}(\mathbf{r}\|\mathbf{m}) \in \mathbb{Z}_q^n$	$\mathbf{a}(\mathbf{r}\|\mathbf{m}) \in R_q = \mathbb{Z}_q[x]/(x^n+1)$
Decommitment string	$(\mathbf{r},\mathbf{m}) \in \{0,1\}^m$	$(\mathbf{r},\mathbf{m}) \in \{0,1\}^m$
Communication complexity	$m + n\lceil\log q\rceil$	$m + n\lceil\log q\rceil$
Homomorphy	-	-

Table 4.4: An overview of the unconditionally hiding schemes from Chapter 3.

[KTX08]	[Xag10]
$q \geq 5793m$	$q > 712m$
$m > 1280\log q$	$m > 512\log q$

Table 4.5: Bounds on the parameters q, m, where $n = 128$, and $E = 3$.

the parameters m, q have the same values in both schemes, then regarding our criteria, both schemes have the same efficiency. Nevertheless, the scheme [Xag10] still outperforms the scheme [KTX08]. This can be justified by considering the public key size. The scheme [Xag10] has at most $m\lceil\log q\rceil$ bits of public key, whereas the public key size of the scheme [KTX08] is at most $nm\lceil\log q\rceil$ bits. Furthermore, if we consider local computations, we maintain the same result, since multiplication over the ring R_q can be performed very efficiently using Fast Fourier Transform.

Remark 4.4. We note that both commitment schemes [KTX08] and [Xag10] are not additively homomorphic. This is because adding two correctly computed commitments yields a commitment of a message, which is not necessarily a bit string. This can affect the output of the verification algorithm of both commitment schemes, since it can reject a correctly computed commitment. As a result, we don't have a post-quantum commitment scheme that is unconditionally hiding and additively homomorphic.

Remark 4.5. We note that we don't have a post-quantum commitment scheme with the perfectly hiding property.

4.2.2 Unconditionally Binding

First we give in Table 4.6 an overview of the unconditionally binding commitment schemes from Chapter 3. All schemes are non-interactive. Therefore, we omit the interaction column.

The scheme	Model	Message type	Computational assumption	Homomorphy	Communication complexity
3.2 [BXH10]	sta.	bit***	LWE	-	$poly(n)$
3.3 [JKPT12]	per.	bit string	xLPN	-	$t+n+\ell^{*}$
3.4 [XXW13]**	per.	integer vector	RLWE	-	$(t+\ell+1)n\lceil \log q \rceil^{*}$

* The parameters ℓ, t, and q are explained below.
** This scheme was also proposed in [BKLP14] with a slight difference.
*** The work of [BLX11] performs bitwise commitment to commit to bit strings.

Table 4.6: An overview of the unconditionally binding schemes from Chapter 3.

Table 4.6 shows that the scheme [BXH10] is very inefficient, since it requires $poly(n)$ bits of communication for committing to only a single bit. This also applies to the extended commitment scheme [BLX11], which performs bitwise commitment for committing to bit strings.

In order to compare the remaining two schemes [JKPT12, XXW13], we examine both schemes more precisely in Table 4.7, where n, n' denote the security parameters of the schemes [JKPT12, XXW13], respectively.

To obtain concrete results for the communication complexity of the commitment schemes from Table 4.7, we have to select for both schemes concrete parameters, which ensure that the computational problem is hard with respect to a particular security level (e.g., 80-bit security), while still keeping parameters as small as possible for performance reasons. To achieve this, it is essential to determine the fastest known algorithms for solving (R)LWE and LPN.

The best known algorithm for solving LPN was recently presented in [GJL14]. In [GJL14], the authors suggested the instance $\text{LPN}_{592,0.125}$ to achieve 80-bit security, i.e., the length of the secret vector is $n = 592$ bits, and the error vector $\mathbf{e} \in \{0,1\}^t$ is chosen from Bernoulli distribution with parameter $\tau = 0.125$.

Recently, the work of [APS15] gathered and presented the available algorithms for solving LWE. The authors of this work stated that in the literature, the hardness of LWE was established only asymptotically, and most algorithms that solve LWE

The scheme	[JKPT12]	[XXW13]
Model	perfectly	perfectly
Interaction	one round	one round
Type of the message $\mathbf{m}$	ℓ-bit string**	$\mathbf{m} \in R_q^{\ell'}$*
Computational assumption	xLPN	RLWE
Parameters t, t'	$t \in O(n+\ell)$	$t' = (\ell'+1) \cdot \omega(\log n')$*
Public key	$\mathbf{A} \xleftarrow{\$} \{0,1\}^{t \times (n+\ell)}$	$\mathbf{A} \xleftarrow{\$} R_q^{t' \times (\ell'+1)}$
Random string	$\mathbf{r} \in \{0,1\}^n$	$r \in R_q$
Commitment string	$\mathbf{A}(\mathbf{r} \| \mathbf{m}) \oplus \mathbf{e} \in \{0,1\}^t$	$\mathbf{A}(r \| \mathbf{m}) + \mathbf{e} \in R_q^{t'}$
Decommitment string	$(\mathbf{r}, \mathbf{m}) \in \{0,1\}^{n+\ell}$	$(r, \mathbf{m}) \in R_q^{\ell'+1}$
Communication complexity	$t + n + \ell$	$(t' + \ell' + 1)n' \lceil \log q \rceil$
Homomorphy	-	-

* $R_q = \mathbb{Z}_q[x]/(\Phi_m(x))$, $\varphi(m) = n'$, $q = poly(n')$, and for efficiency, ℓ' and t' are chosen such that $\ell' \in O(1)$, and $t' = \omega(\log n')$.

** To perform efficiency, ℓ is chosen such that $\ell \approx n$.

Table 4.7: Comparison between the scheme [JKPT12] and [XXW13].

have no tight closed formulae for expressing their complexities. Their work also indicates that there is no particular algorithm, which always outperforms all others, and the performance of the algorithms depends on the selected parameters. On the other hand, in the work of [LPR13], where RLWE was defined, the authors stated that the asymptotically fastest known algorithms for ideal lattices perform essentially no better than their generic algorithms for general lattices.

Therefore, performing our comparison requires selecting parameters for the commitment scheme [XXW13], which assure that the best attack strategy for solving LWE takes at least $2^{n'}$ operations, for the lattice dimension n' that achieves 80-bit security. The results of [APS15] shows that the dimension $n' = 128$ does not provide 80-bit security. Moreover, RLWE requires n' to be a power of two. Thus we set the dimension to $n' = 256$, which should achieve 80-bit security, according to the results of [APS15].

After estimating concrete security parameters on both commitment schemes for some security level, we have to choose the remaining parameters of the schemes as small as possible according to their bounds given in Table 4.7. Therefore, we set $\ell = n$, so that $t = 2n$, and $\ell' = 1$, $t' = 1 + \lfloor \log n' \rfloor$. We obtain two minimal sets of parameters given in Table 4.8.

The sets of parameters given in Table 4.8 show that the scheme [JKPT12] requires

[JKPT12]	[XXW13]
$n = 592$	$n' = 256$
$\|\mathbf{m}\| = 592$	$\|\mathbf{m}\| = 256\lceil \log q \rceil$
$t = 1184$	$t' = 9$

Table 4.8: Minimal sets of parameters for the commitment schemes [JKPT12, XXW13], for 80-bit security.

$t + n + \ell = 2368$ bits of communication, whereas the scheme [XXW13] requires $(t' + \ell' + 1)n'\lceil \log q \rceil = 2816\lceil \log q \rceil$ bits, which are more than 2368, for any $q = poly(n')$. Finally, we conclude that the scheme [JKPT12] has lower communication complexity than that of the scheme [XXW13].

Remark 4.6. We point out that even if we commit to a message of the same bit length in both commitment schemes, we obtain the same result. This can be seen by setting $|\mathbf{m}| = \ell = n'\lceil \log q \rceil = 256\lceil \log q \rceil$, which implies $t = n + \ell = n + n'\lceil \log q \rceil = 592 + 256\lceil \log q \rceil$. The resulting communication complexity of the scheme [JKPT12] is $t + n + \ell = 1184 + 512\lceil \log q \rceil$ bits, which is still smaller than that of the scheme [XXW13], for any $q = poly(n')$.

Another fact that we can consider is that LWE is a natural extension of the LPN problem, i.e., the special case $q = 2$ of LWE corresponds to the LPN problem. According to [Reg10], the security of (R)LWE requires q to be somewhat large. Therefore, at least the factor $\lceil \log q \rceil$ makes the communication complexity of the scheme [XXW13] larger than that of the scheme [JKPT12].

Remark 4.7. We note that the commitment scheme [JKPT12] is not additively homomorphic. This is due to the fact that the error vector $\mathbf{e} \in \{0,1\}^t$ from the commitment string must have an exact Hamming weight. Hence, adding two error vectors of some Hamming weight ω does not necessarily yield a vector of the same Hamming weight ω. This can affect the output of the verification algorithm of the scheme. A relatively similar reason applies to the commitment scheme [XXW13]. This is due to its verification algorithm, which verifies whether the norm of the error vector is upper-bounded by the predetermined bound β of the error distribution. However, a limited number of additions could be performed by selecting error vectors that have norm much smaller than β. This would not affect the output of the verification algorithm of the scheme. As a result, we don't have a post-quantum commitment scheme that is unconditionally binding and additively homomorphic.

4.3 Comparing All Together

In this section we compare the results of Section 4.1 with those of Section 4.2. More precisely, we compare the resulting unconditionally hiding commitment schemes from Subsection 4.1.1 and 4.2.1 with each other, and the resulting unconditionally binding commitment schemes from Subsection 4.1.2 and 4.2.2 with each other. We recall that our comparison considers interaction and communication complexity.

4.3.1 Unconditionally Hiding

The results of Subsection 4.1.1 and 4.2.1 are the statistically hiding commitment schemes [DPP98, Xag10], respectively. The former scheme employs any family of collision resistant hash function (CRHFs), whereas the latter scheme is based on the Ring-SIS problem.

Before comparing both commitment schemes, it is substantial to consider works that improve the parameters of the (Ring-) SIS problem, while preserving its hardness. The most recent result was shown in [MP13]. This result shows that the SIS problem retains its hardness for moduli q nearly equal to the solution bound β. More precisely, the work of [MP13] gives a polynomial-time reduction from $\mathrm{SIVP}^{p}_{\gamma\cdot\omega(\sqrt{\log n})}$ in the worst case on n-dimensional lattices to $\mathrm{SIS}^{p}_{q,m,\beta}$ on the average case, where $q \geq \beta \cdot n^{\Omega(1)}, m = poly(n), \gamma = \max\{1, \beta/n^{\Omega(1)}\} \cdot O(\beta\sqrt{n})$, and $p \geq 1$. In [MP13], the authors stated that the lower bound given for q is essentially optimal, since the problem is trivially easy for $q \leq \beta$.

The result of [MP13] can be applied to the commitment scheme [Xag10], whose binding property is based on the ring variant of SIS (Ring-SIS) problem. This requires setting n to be a power of two, m to be a multiple of n, and q to be a prime such that $2n$ divides $q-1$. However, the parameter m must be large enough to ensure the statistically hiding property.

The commitment scheme [DPP98] uses any CRHF f^* that maps bit strings of arbitrary lengths to bit strings of length $n'+1$, for a security parameter n'. According to [DPP98], such a function can be constructed from any CRHF f with given input length m', and output length $n' < m'$, using the Merkle-Damgård technique. Therefore, the hash size n' of the CRHF f corresponds to the security parameter of the commitment scheme [DPP98]. Unlike the commitment scheme [Xag10], the scheme [DPP98] manages the statistically hiding property by using some family of 2-universal hash functions. We can use one of the 2-universal hashing constructions

given in Example 2.1 and 2.2. However, since the utilized family of 2-universal hash functions affects the communication complexity of the commitment scheme [DPP98], we adopt a family given in [Sho09]. Describing any function from this family requires less bits than describing any function from the family given in either Example 2.1 or Example 2.2.

Remark 4.8. We point out that there may be in the literature 2-universal hash functions, whose description requires less bits than describing any function from the family given in [Sho09]. This would make the communication complexity of the commitment scheme [DPP98] even lower.

The following example describes the family of 2-universal hash functions which we adopt from [Sho09].

Example 4.1 (2-Universal Hashing Construction III)**.** For a positive integer ℓ, consider the family $\mathcal{H} = \{H_{\mathbf{a}_1,\ldots,\mathbf{a}_{\ell-1}} : \mathbf{a}_1,\ldots,\mathbf{a}_{\ell-1} \in \mathbf{GF}(2^n)\}$.
Each tuple $(\mathbf{a}_1,\ldots,\mathbf{a}_{\ell-1})$ describes a function $H_{\mathbf{a}_1,\ldots,\mathbf{a}_{\ell-1}} : \mathbf{GF}(2^n)^\ell \longrightarrow \mathbf{GF}(2^n)$ such that $H_{\mathbf{a}_1,\ldots,\mathbf{a}_{\ell-1}}(\mathbf{x}_0,\ldots,\mathbf{x}_{\ell-1}) = \mathbf{x}_0 + \mathbf{a}_1\mathbf{x}_1 + \ldots + \mathbf{a}_{\ell-1}\mathbf{x}_{\ell-1}$, where each n-bit string $\mathbf{x}_i$ is interpreted as an element in $\mathbf{GF}(2^n)$, for all $i = 0,\ldots,\ell-1$. Any function from this family can be described using $n(\ell-1)$ bits.

The commitment scheme [DPP98] requires a family of 2-universal hash functions that maps t-bit strings to $(n'+1)$-bit strings, where $t = 3(n'+1)$. In order to use a function H_k from the 2-universal family $\mathcal{H}$ given in Example 4.1, we adjust the function to $H_{\mathbf{a}_1,\mathbf{a}_2} : \mathbf{GF}(2^{n'+1})^3 \longrightarrow \mathbf{GF}(2^{n'+1})$, where $k = (\mathbf{a}_1,\mathbf{a}_2) \xleftarrow{\$} \mathbf{GF}(2^{n'+1})^2$. Therefore, we need only $2(n'+1)$ bits to describe the function $H_{\mathbf{a}_1,\mathbf{a}_2}$.

Table 4.9 gives a brief overview of the commitment schemes [DPP98, Xag10], where n', n denote their security parameters, respectively.

The scheme	2.3.3 [DPP98]	2 [Xag10]
Model	statistically	statistically
Interaction	one round	one round
Type of the message $\mathbf{m}$	ℓ-bit string	$(m/2)$-bit string
Computational assumption	CRHF	Ring-SIS
Communication complexity	$6(n'+1)+\ell$	$m + n\lceil \log q \rceil$
Homomorphy	-	-

Table 4.9: Comparison between the commitment schemes [DPP98] and [Xag10].

Determining which commitment scheme has lower communication complexity depends on the choice of the parameters for some security level. The parameters of

the scheme [Xag10] have to be chosen according to [MP13] such that the Ring-SIS problem is hard for some security level. Furthermore, the parameter m has to be chosen large enough in order to ensure some negligible statistical gap. However, establishing a concrete value for the security parameter n' of the commitment scheme [DPP98] for some security level, depends on the hash size of the employed CRHF. Therefore, we give in the following an upper bound on the security parameter n' (or the hash size n' of the utilized CRHF), under which the commitment scheme [DPP98] outperforms the scheme [Xag10] for any security level κ. This upper bound assumes that the message being committed to is of the same length in both schemes, i.e., $\ell = m/2$ (see Table 4.9).

We let $n(\kappa)$ denote the security parameter of the scheme [Xag10] and $n'(\kappa)$ the security parameter of the scheme [DPP98]. The parameter $n'(\kappa)$ is also the hash size of the employed CRHF that achieves a security level of κ for collision resistance. Furthermore, we use the family of 2-universal hash functions given in Example 4.1 for the scheme [DPP98]. Then, according to Table 4.9, the scheme [DPP98] requires $6(n'(\kappa) + 1) + m/2$ bits of communication, whereas the scheme [Xag10] requires $m + n(\kappa) \cdot \lceil \log q \rceil$ bits. Therefore, the scheme [DPP98] has communication complexity lower than that of the scheme [Xag10], if it holds that

$$6(n'(\kappa) + 1) + m/2 < m + n(\kappa) \cdot \lceil \log q \rceil \ .$$

This implies

$$n'(\kappa) < (2n(\kappa) \cdot \lceil \log q \rceil + m - 12)/6 \ . \tag{4.1}$$

We conclude that our result of comparing the commitment scheme [DPP98] with the scheme [Xag10] depends on the upper bound on the hash size of the utilized CRHF in the scheme [DPP98] which we established in Inequation 4.1.

Remark 4.9. We note that we did not obtain a post-quantum commitment scheme that is unconditionally hiding and additively homomorphic.

Remark 4.10. We point out that both commitment schemes we compared in this subsection provide statistically hiding commitments. If a post-quantum commitment scheme with the perfectly hiding property is required, then the only commitment scheme that can be used is the scheme proposed in [Dam89]. This scheme employs some Σ-protocol with the perfect zero-knowledge property for some hard relation. In addition, the hard relation must have the property that it is easy to verify membership in the language defined by this relation.

4.3.2 Unconditionally Binding

We give in Table 4.10 a brief overview of the resulting unconditionally binding commitment schemes from Subsection 4.1.2 and 4.2.2.

The scheme	2.1.1 [Nao91]	3.3 [JKPT12]
Model	statistically	perfectly
Interaction	two rounds	one round
Type of the message	ℓ-bit string, $\ell \geq O(n)$	ℓ-bit string*
Computational assumption	PRG	xLPN
Communication complexity	$O(\ell)$	$t + n + \ell,\ t \in O(n + \ell)$
Homomorphy	-	-

* To perform efficiency, ℓ is chosen such that $\ell \approx n$.

Table 4.10: Comparison between the scheme [Nao91] and [JKPT12].

Table 4.10 shows that the communication complexity of both commitment schemes are relatively close. The commitment scheme [Nao91] uses some pseudorandom generator (PRG). On the other hand, the definition of the decisional xLPN problem already implies that xLPN samples are pseudorandom. Hence, the commitment value of the scheme [JKPT12] is pseudorandom as well. In the commitment scheme [Nao91], the bit length ℓ of the message being committed to is at least linear in the security parameter n, whereas it is arbitrary in the scheme [JKPT12]. In order to commit to a message of the same length, we set $\ell \in O(n)$. This implies that both schemes require $O(n)$ bits of communication.

Regarding the interaction, it is obvious that the scheme [JKPT12] outperforms the scheme [Nao91], since the former is non-interactive, whereas the latter is interactive with two rounds. We conclude that the commitment scheme [JKPT12] is more efficient than [Nao91].

Remark 4.11. It is also worth to point out that the binding property of the scheme [JKPT12] holds perfectly, whereas it holds statistically in the construction [Nao91].

Remark 4.12. We note that we did not obtain a post-quantum commitment scheme that is unconditionally binding and additively homomorphic.

Conclusion and Future Work

In this thesis we provided a survey on unconditionally hiding and unconditionally binding commitment schemes that run on conventional computers and are believed to be secure even against quantum computers. The survey included commitment schemes that can be obtained from general frameworks that employ cryptographic primitives or protocols. In addition, the survey included commitment schemes that are directly based on computational problems that are believed to be secure even under quantum attacks. We have seen that most general constructions use additional tools in their framework such as hard-core predicates, interactive hashing protocols, and 2-universal hash functions.

In addition to the survey, we provided a three-phase comparison of the commitment schemes included in the survey. The purpose of this comparison is to obtain two post-quantum commitment schemes, one that is unconditionally hiding, and one that is unconditionally binding. Both desired schemes are required to be practical and efficient with respect to interaction and communication complexity. The first phase of the comparison involved the general constructions of commitment schemes from the survey, whereas the second phase involved the concrete commitment schemes. The third phase involved the results of the first and second phase.

Regarding unconditionally binding commitments, the result of our comparison is the non-interactive perfectly binding commitment scheme presented in [JKPT12]. This scheme is based on the learning parity with noise (LPN) problem.

The unconditionally hiding commitment schemes involved in the third phase of the comparison are the non-interactive statistically hiding commitment schemes proposed in [DPP98] and [Xag10]. The construction presented in [DPP98] uses any family of collision resistant hash functions (CRHFs) in addition to any family of 2-universal hash functions. The scheme presented in [Xag10] is the ring variant of the scheme introduced in [KTX08]. It is based on the ring small integer solution (Ring-SIS) problem which is the source of ideal-lattice-based hash functions. For comparing both commitment schemes, we used the most recent results shown in [MP13] to improve the parameters of the commitment scheme proposed in [Xag10]. Moreover, we improved the communication complexity of the commitment scheme proposed in [DPP98] by adopting a family of 2-universal hash functions given in [Sho09]. Describing any function from this family requires less bits than describing any function from the 2-universal family suggested in [DPP98] for the efficiency of the scheme. We concluded that determining the commitment scheme with the lowest communication complexity depends on the hash size of the CRHF utilized

in the scheme proposed in [DPP98], for some security level. We established an upper bound on this hash size, under which the commitment scheme presented in [DPP98] outperforms the scheme presented in [Xag10], for any security level. In addition, we concluded that the only commitment scheme that provides perfectly hiding commitments is the scheme proposed in [Dam89]. This scheme employs some Σ-protocol with the perfect zero-knowledge property for some hard relation. In addition, the hard relation must have the property that it is easy to verify membership in the language defined by this relation.

For applications that require additively homomorphic commitments, we concluded that the commitment scheme presented in [XXW13] is the only scheme that could be additively homomorphic to a limited degree. This scheme is non-interactive, perfectly binding, and is based on the ring learning with errors (RLWE) problem. This shows the lack of post-quantum commitment schemes with homomorphic properties.

Regarding our limited result related to statistically hiding commitments, we see that it is useful to search for practical post-quantum CRHFs. This allows to investigate if their hash sizes satisfy the upper bound, which determines if the commitment scheme introduced in [DPP98] outperforms the scheme introduced in [Xag10]. In addition to CRHFs, it is also useful to investigate if there exist 2-universal hash functions that are more efficient than those considered in this work. This would improve the established upper bound as well. Furthermore, we believe that additional research is required regarding post-quantum commitment schemes with the perfectly hiding property. This is significant, since there exists only a general construction that provides perfectly hiding commitments using Σ-protocols with special conditions. Finally, we think that further research regarding post-quantum commitment schemes with homomorphic properties is required as well. This is due to the lack of such commitment schemes.

References

[AC02] Mark Adcock and Richard Cleve. A quantum Goldreich-Levin theorem with cryptographic applications. In *STACS 2002*, pages 323–334. Springer, 2002.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM, 1996.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. http://eprint.iacr.org/.

[BB84] CH Bennett and G Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proc. IEEE International Conference on Computers Systems and Signal Processing*, pages 175–179, 1984.

[BBCM95] Charles H Bennett, Gilles Brassard, Claude Crépeau, and Ueli M Maurer. Generalized privacy amplification. *Information Theory, IEEE Transactions on*, 41(6):1915–1923, 1995.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BIKM99] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. One-way functions are essential for single-server private information retrieval. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 89–98. ACM, 1999.

[BKLP14] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. Cryptology ePrint Archive, Report 2014/889, 2014. http://eprint.iacr.org/.

[Blu82] Manuel Blum. Coin flipping by telephone: A protocol for solving impossible problems. *Advances in Cryptology-A Report on CRYPTO'81*, 1982.

[BLX11] Zeng Bing, Chen Liang, and Tang Xueming. A perfectly binding commitment scheme against quantum attacks. *Citeseer*, 2011.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing*, 13(4):850–864, 1984.

[BXH10] Zeng Bing, Tang Xueming, and Chingfang Hsu. A framework for fully-simulatable h-out-of-n oblivious transfer. *arXiv preprint arXiv:1005.0043*, 2010.

[CD04] Jordi Castellà-Roca and Josep Domingo-Ferrer. A non-repudiable bit-string commitment scheme based on a public-key cryptosystem. In *International Conference on Information Technology: Coding and Computing (ITCC'04), Volume 2, April 5-7, 2004, Las Vegas, Nevada, USA*, pages 778–780, 2004.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology—Crypto 2001*, pages 19–40. Springer, 2001.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[CGKS98] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[Cra96] Ronald John Fitzgerald Cramer. *Modular Design of Secure, yet Practical Cryptographic Protocols*. PhD thesis, University of Amsterdam, 1996.

[CW77] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112. ACM, 1977.

[Dam89] Ivan Bjerre Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In *Advances in Cryptology—CRYPTO'89 Proceedings*, pages 17–27. Springer, 1989.

[Dam90] Ivan Bjerre Damgård. A design principle for hash functions. In *Advances in Cryptology—CRYPTO'89 Proceedings*, pages 416–427. Springer, 1990.

[Dam02] Ivan Damgard. On Σ-protocols. *Lecture notes for CPT*, 2002.

[DC03] Giovanni Di Crescenzo. Equivocable and extractable commitment schemes. In *Security in Communication Networks*, pages 74–87. Springer, 2003.

[DDN03] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM review*, 45(4):727–784, 2003.

[DFS04] Ivan Damgård, Serge Fehr, and Louis Salvail. Zero-knowledge proofs and string commitments withstanding quantum attacks. In *Advances in Cryptology–CRYPTO 2004*, pages 254–272. Springer, 2004.

[DMS00] Paul Dumais, Dominic Mayers, and Louis Salvail. Perfectly concealing quantum bit commitment from any quantum one-way permutation. In *Advances in Cryptology—EUROCRYPT 2000*, pages 300–315. Springer, 2000.

[DPP97] Ivan B Damgård, Torben P Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10(3):163–194, 1997.

[DPP98] Ivan B Damgard, Torben P Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *Information Theory, IEEE Transactions on*, 44(3):1143–1151, 1998.

[Eve83] Shimon Even. A protocol for signing contracts. *ACM SIGACT News*, 15(1):34–39, 1983.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology-AUSCRYPT*, pages 244–251. Springer, 1992.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 3, pages 236–241, 1996.

[GJL14] Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN using covering codes. In *Advances in Cryptology–ASIACRYPT 2014*, pages 1–20. Springer, 2014.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–189, 1996.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304. ACM, 1985.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

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

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.

[HHK+05] Iftach Haitner, Omer Horvitz, Jonathan Katz, Chiu-Yuen Koo, Ruggero Morselli, and Ronen Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. In *Advances in Cryptology–EUROCRYPT 2005*, pages 58–77. Springer, 2005.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in Cryptology—CRYPTO'96*, pages 201–215. Springer, 1996.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[HPS08] Jeffrey Hoffstein, Jill Catherine Pipher, and Joseph H Silverman. *An introduction to mathematical cryptography*. Springer, 2008.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *Theory of Cryptography*, pages 445–456. Springer, 2005.

[JKPT12] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *Advances in Cryptology–ASIACRYPT 2012*, pages 663–680. Springer, 2012.

[Jus72] Jørn Justesen. Class of constructive asymptotically good algebraic codes. *Information Theory, IEEE Transactions on*, 18(5):652–656, 1972.

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. *IACR Cryptology ePrint Archive*, 2005:328, 2005.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *Advances in Cryptology-ASIACRYPT 2008*, pages 372–389. Springer, 2008.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International Palo Alto, 1979.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *Automata, Languages and Programming*, pages 144–155. Springer, 2006.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In *Fast Software Encryption*, pages 54–72. Springer, 2008.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):43, 2013.

[Mer90] Ralph C Merkle. One way hash functions and DES. In *Advances in Cryptology—CRYPTO'89 Proceedings*, pages 428–446. Springer, 1990.

[Mic11] Daniele Micciancio. The geometry of lattice cryptography. In *Foundations of security analysis and design VI*, pages 185–210. Springer, 2011.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *Advances in Cryptology–CRYPTO 2013*, pages 21–39. Springer, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

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

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, 1998.

[NV06] Minh-Huyen Nguyen and Salil Vadhan. Zero knowledge with efficient provers. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 287–295. ACM, 2006.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43. ACM, 1989.

[OVY91] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Fair games against an all-powerful adversary. *Advances in Computational Complexity Theory*, 13:155–169, 1991.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91*, pages 129–140. Springer, 1992.

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In *SOFSEM 2012: Theory and Practice of Computer Science*, pages 99–114. Springer, 2012.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[Reg10] Oded Regev. The learning with errors problem. *Invited survey in CCC*, 2010.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394. ACM, 1990.

[Sho97] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[SRA81] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman. Mental poker. *The Mathematical Gardner*, pages 37–43, 1981.

[Ste96] Jacques Stern. A new paradigm for public key identification. *Information Theory, IEEE Transactions on*, 42(6):1757–1768, 1996.

[TPY08] Chunming Tang, Dingyi Pei, and Zheng-an Yao. 5-round computational zero-knowledge proof with negligible error probability for any NP from any one-way permutation. In *Electronic Commerce and Security, 2008 International Symposium on*, pages 407–411. IEEE, 2008.

[WP89] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. *J.-J. Quisquater and J. Vandewalle, editors, Advances in Cryptology—EUROCRYPT*, 89:690, 1989.

[Xag10] D Keita Xagawa. *Cryptography with Lattices*. PhD thesis, Tokyo Institute of Technology, Tokyo, Japan, 2010. Available at http://xagawa.net/pdf/2010Thesis.pdf.

[XXW13] Xiang Xie, Rui Xue, and Minqian Wang. Zero knowledge proofs from Ring-LWE. In *Cryptology and Network Security*, pages 57–73. Springer, 2013.

[Yam13] Tomoyuki Yamakami. A non-interactive quantum bit commitment scheme that exploits the computational hardness of quantum state distinction. *arXiv preprint arXiv:1309.0436*, 2013.

Index

Symbols

Σ-protocol 56
q-ary lattice 17
1-out-of-2 binding 34
2-universal hash function 44

A

approximable-preimage-size one-way function 37

B

binding commitments 12

C

coin flipping problem 14
collision resistant hash function . . . 24
commitment scheme 11
CRT representation 19
cyclotomic polynomial 19

D

determinant of a lattice 16
digital auctions 13
digital signature scheme 25
discrete Gaussian distribution 18
dual lattice 16

E

error-correcting code 30
exact learning parity with noise . . . 71
expansion factor 19

F

fail-stop signature scheme 46
full rank lattice 15
fundamental parallelepiped 16

G

gap version of shortest vector problem 20
Gaussian function 18
Goldreich-Levin predicate 38

H

hard relation 57
hard-core predicate 38
hiding commitments 13
homomorphic commitment scheme 51
homomorphic commitments 13
homomorphic one-way commitment scheme 52

I

ideal 18
ideal lattice 18
ideal-lattice-based hash function . . 65
integer lattice 15
interactive hashing 39

L

lattice 15
lattice basis 15
lattice dimension 15
lattice-based hash function 64
learning parity with noise problem 22
learning with errors problem 21
length-preserving function 23

M

Merkle-Damgård technique 67
minimal polynomial 19
minimum distance 17

N

negligible function 10
non-malleable commitment scheme 27
normal distribution 18

O

one-way function 22
one-way permutation 23
overwhelming probability 10

P

pairwise independent hash function 34
polynomial factor ring 18
post-quantum commitment scheme 62
primitive m-th root of unity 19
principal ideal 18
private information retrieval 59
pseudorandom generator 23
public-key encryption 24

R

regular one-way function 37
ring variant of learning with errors problem 21
ring variant of small integer solution problem 20

S

shortest independent vectors problem 20
shortest vector problem 19
small integer solution problem 20
standard deviation 18
statistical distance 10
statistically close distributions 10
successive minima 16

T

trapdoor commitment scheme 27
two-phase commitment scheme ... 33

U

universal one-way hash function .. 34
universally composable commitment scheme 27

Z

zero-knowledge arguments 14
zero-knowledge proofs 14