AARHUS UNIVERSITY

COMPUTER SCIENCE

MASTER'S THESIS

Statistically Secure Sigma Protocols with Abort

Author:
Anders Fog BUNZEL
(20112293)

Supervisor:
Ivan Bjerre DAMGÅRD

September 2016

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

AARHUS UNIVERSITY

Abstract

Statistically Secure Sigma Protocols with Abort

by Anders Fog BUNZEL

An efficient way that one part, called the prover, can identify himself towards another part, called the verifier, is with a sigma protocol that satisfies completeness, special soundness and special honest verifer zero-knowledge (sHVZK). However, depending on the problem that the security is based on it may be insecure for the prover to use a sigma protocol since he may not be able to hide his secret for a malicious verifier. One such example is if the security is based on a lattice problem like the shortest vector problem. The literature has proposed an identification scheme that use a technique called abort, which deals with this problem. But an identification scheme is weaker than a sigma protocol since a sigma protocol can be used as a signature scheme, a zero-knowledge protocol, a commitment scheme or even an identification scheme. In this thesis, we present a statistically secure sigma protocol with abort that satisfies statistical completeness, statistical special soundness and computational sHVZK. We find that the statistically secure sigma protocol with abort is a nice-to-have protocol, since the security of traditionally sigma protocols are either based on the prime factorization problem or the discrete logarithm problem. And if these problems someday are proven to be easy, it's important to have other kind of problems like lattice problems that we can base the security on.

Acknowledgements

I would first like to thank my thesis supervisor Prof. Ivan Bjerre Damgård for his excellent help and great ideas. The door to Prof. Damgård was always open when I ran into trouble or had a question about my research or writing. I would also like to thank my office roommates Tobias Stenby Brixen and Troels Thorsen for talking about things other than just our theses. In addition, I would to thank Tobias for his technical skills in LaTex, ideas and input.

Anders Fog Bunzel,
Aarhus, September 3, 2016.

Contents

Abstract **i**

Acknowledgements **ii**

1 Introduction **1**
1.1 Chapter Overview 2

2 Preliminaries **4**
2.1 Notations 4
2.2 The Chernoff-Hoeffding Bound 4
2.2.1 The Chernoff-Hoeffding Bound with Limited Independence 5
2.3 Lattices 6
2.4 Encoding Schemes 7
2.4.1 Secret Sharing using Encoding Schemes 8
2.5 Negligibility 9
2.6 Indistinguishability 9
2.7 Commitment Schemes 10
2.8 Interactive Proof Systems 12
2.8.1 Proof of Knowledge Systems 13
2.9 Zero-Knowledge Protocols 14
2.10 Sigma Protocols 16
2.10.1 Statistically Secure Sigma Protocols 18
2.10.2 sHVZK vs. HVZK 18
2.11 The Random Oracle Model 19

3 A General Framework for Protocols with Abort **21**
3.1 The General Framework with Abort 21
3.1.1 Proof of THEOREM 3.1 23
3.2 Comparison of the Aborting and Non-Aborting Version of the General Framework 29

4 Applications of the General Framework **33**
4.1 Protocols based on the General Framework 33
4.2 A Statistically Secure Sigma Protocol in the Standard Model based on the General Framework with Abort 34
4.3 A Statistically Secure Non-Interactive Sigma Protocol in the Random Oracle Model based on the General Framework with Abort 40

5 Conclusion **43**

List of Protocols **44**

Bibliography **45**

Chapter 1

Introduction

This thesis is a continuous work of [12], and hence small parts of this thesis has previously been published.

A sigma protocol as defined in [9] is an interactive protocol between two parties where one party, called the prover, tries to convince another party, called the verifier, that he know some piece of information, which enable him to prove that a certain statement is true. One such example is that he tries to convince the verifier about that he know the value of a secret key corresponding to a given public key. To be more precise, then the prover claims to know the value of the witness w to the problem x in the relation R, which is also denoted as $(w, x) \in R$. The problem x is based on the hardness of some computational problem defined in the group G and consists of the description of an additive homomorphic function f and the element $y = f(w)$.

In the first step of a sigma protocol chooses the prover, say P, some randomness r from the order of the group G and sends $a = f(r)$ to the verifier, say V. V then sends a t-bit challenge e to P, who use the witness w, the randomness r and the challenge e to compute the response z, e.g. $z = r + e \cdot w \bmod |G|$ where $|G|$ is the order of G, which he sends to V. Finally, V verifies that (a, e, z) is an accepting conversation for the problem x: if it's he outputs accept and otherwise he outputs reject.

A sigma protocol satisfies completeness, special soundness and special honest verifier zero-knowledge. The completeness property ensures that if P is honest and they both follows the protocol P can always convince V to output accept. Similar, the special soundness property ensures that if P is malicious, i.e. he don't know the value of a correct witness to the problem x, he is only able of convincing V to output accept with probability negligible in the length of the challenge e. Finally, the special honest verifier zero-knowledge property ensures that the only information V learns after having seen all messages sent during the protocol is that P know the value of a correct witness to the problem x in the relation R and nothing else.

The reason why P has to choose the randomness r is to hide the value of the witness w in the response z. If G is some finite group P would just choose r uniformly at random from the order of G, but if the order of G is unknown this is an infeasible solution. One example is Girault's protocol [1, 3] where $G = \mathbb{Z}_n$ with $n = p \cdot q$ for the primes p and q, which both are unknown to P and V. And hence, they don't know the order of G, which is $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid \gcd(x, \phi(n)) = 1\}$, because they can't compute Euler's phi function $\phi(n) = (p-1) \cdot (q-1)$. Instead, P can choose w and r from intervals where the interval that r is chosen from is much larger compared to the one

that w is chosen from. The value of w is now hidden in e.g. $z = r + b \cdot w$ due to the fact that adding a small integer (the witness w) to a large integer (the randomness r) doesn't have much influence on the result, i.e. we can think of the response z as $z = r$. Notice that the challenge is only a single bit b.

We also have to use the above approach if the group G is infinite, e.g. $G = \mathbb{Z}$. However, in some cases is this insecure. One such example is if the security is based on a lattice problem like the shortest vector problem as e.g. in Lyubashevsky's protocol [7]. This protocol use the infinite group $G = \mathbb{Z}^n$ where $n \geq 1$ is the vector length. Choosing r from a much larger interval than w implies that a possible malicious verifier V^* would be able of computing a correct witness w^* such that $(w^*, x) \in R$ because he only has to find a large preimage of y. A solution to this problem is for P to choose r from an interval that is only a small factor larger than the one w is chosen from and use a technique called abort: In the third step of the protocol P only reveals the response z to V if it inside some interval I and aborts otherwise. Now V^* has a hard time because he has to find a small preimage of y.

Our contribution. The contribution of this work is a statistically secure sigma protocol with abort, i.e. a sigma protocol using the above described abort technique that satisfies statistical completeness, statistical special soundness and computational special honest verifier zero-knowledge.

1.1 Chapter Overview

Chapter 2. In the preliminary chapter we introduce the notations and definitions used in the rest of the thesis. The notations and definitions are based on the work of [9, 11]. We start with a few mathematical definitions such as lattice and the Chernoff-Hoeffding bound including the extended version where the independence of the random variables are limited. Then we will introduce some basic cryptography definitions, namely negligibility and indistinguishability. From thereof we define some more advanced definitions such as encoding schemes, secret sharing, commitment schemes, interactive proof systems, zero-knowledge protocols, sigma protocols and the random oracle model.

Chapter 3. In this chapter we present a general framework for protocols with abort and prove that it satisfies completeness with abort, statistical special soundness and computational special honest verifier zero-knowledge. The general framework is on a similar form as a sigma protocol, but where the challenge that the verifier sends to the prover is only a single bit. However, we don't specify what should happen if the prover chooses to abort in the framework, and hence it's only a building block from which we can build e.g. a statistically secure sigma protocol. Finally we present a non-aborting version of the general framework and compare it to the aborting version. This comparison shows how important the aborting technique is security-wise when we use lattice.

Chapter 4. In this chapter we first present two applications of the general framework: Girault's protocol [1, 3] and Lyubashevsky's lattice-based protocol [7]. Then we present a statistically secure sigma protocol, which is

build upon the general framework with abort and prove that it satisfies statistical completeness, statistical special soundness and computational special honest verifier zero-knowledge. Finally we present a non-interactive version of the statistically secure sigma protocol that is only secure in the random oracle model. In contrast to the interactive version can the non-interactive version achieve perfect completeness. However, this come at the expense of the efficiency.

Chapter 5. In the final chapter we present our conclusion and discuss possible future work.

Chapter 2

Preliminaries

2.1 Notations

Throughout this thesis we will use consistent notations based on [9, 11]. We will use a square $\square$ to indicate the end of a proof, a diamond $\lozenge$ to indicate the end of an example and a triangle $\triangle$ to indicate the end of a remark. The abbreviation "iff" is used for "if and only if" and $x \in_R S$ means that x is chosen uniformly at random from the set S.

We define a vector of length $n \geq 1$ as $\vec{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{Z}^n$ and a vector of vectors as $\hat{v} = (\vec{v_1}, \vec{v_2}, \ldots, \vec{v_m})$ for some $m \geq 1$. The infinity norm of $\vec{v}$ is defined as $\|\vec{v}\|_\infty = \max_i |v_i|$ for $i = 1, 2, \ldots, n$ and the infinity norm of $\hat{v}$ as $\|\hat{v}\|_\infty = \max_j \|\vec{v_j}\|_\infty$ for $j = 1, 2, \ldots, m$.

2.2 The Chernoff-Hoeffding Bound

Let X and Y be two random variables. We say that X and Y are independent if knowing the value of one of the variables gives no additional information about the other variable. More formally, X and Y are independent if and only if $\Pr[X = x \mid Y = y] = \Pr[X = x]$ for all possible values of x and y.

The Chernoff-Hoeffding bound says that the outcome of a random experiment repeated many times independently is likely to be close to its expected outcome.

DEFINITION 2.1. Let $X_1, \ldots, X_n$ be independent random variables where $0 \leq X_i \leq 1$ for all i. Let $X = \sum_{i=1}^n X_i$ and $\mu = \sum_{i=1}^n \mathbb{E}[X_i]$ where $\mathbb{E}[X_i]$ is the expected outcome of X_i. Then for any $0 < \epsilon \leq 1$ is:

$$\Pr[X \leq (1-\epsilon) \cdot \mu] \leq \exp\left(-\frac{\epsilon^2 \cdot \mu}{2}\right)$$

$$\Pr[X \geq (1+\epsilon) \cdot \mu] \leq \exp\left(-\frac{\epsilon^2 \cdot \mu}{3}\right)$$

where $\exp(x) = e^x$.

EXAMPLE 2.2. Consider $n = 100$ independently coin flips X_i using a fair coin. Let $X_i = 1$ if coin i turns up head and $X_i = 0$ otherwise. Now, $X = \sum_{i=1}^n X_i$ denotes the actually number of heads in the experiment and $\mu(n) = \sum_{i=1}^n \mathbb{E}[X_i] = 100 \cdot \frac{1}{2} = 50$ the expected number of heads

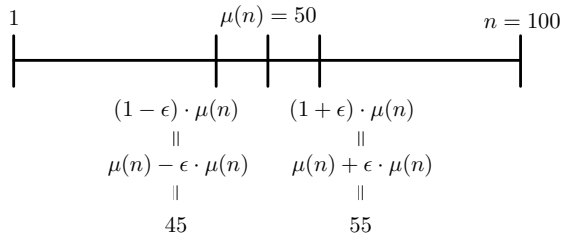

FIGURE 2.1

in the experiment. Say, that we are e.g. interested in the probability that the actually outcome X of the experiment deviates with less/more than $\epsilon \cdot \mu(n) = \frac{1}{10} \cdot 50 = 5$ from the expected outcome $\mu(n) = 50$. See FIGURE 2.1 for an illustration. Using the lower tail Chernoff-Hoeffding bound we have that:

$$\begin{aligned}\Pr[X \leq (1-\epsilon)\cdot\mu(n)] &\leq \exp\left(-\frac{\epsilon^2\cdot\mu(n)}{2}\right) \\ &= \exp\left(-\frac{\left(\frac{1}{10}\right)^2\cdot\frac{n}{2}}{2}\right) \\ &= \exp\left(-\frac{n}{400}\right)\end{aligned}$$

and using the upper tail Chernoff-Hoeffding bound we have that:

$$\begin{aligned}\Pr[X \geq (1+\epsilon)\cdot\mu(n)] &\leq \exp\left(-\frac{\epsilon^2\cdot\mu(n)}{3}\right) \\ &= \exp\left(-\frac{\left(\frac{1}{10}\right)^2\cdot\frac{n}{2}}{3}\right) \\ &= \exp\left(-\frac{n}{600}\right)\end{aligned}$$

In other words, the probability that the actually outcome X deviates with less/more than $\epsilon \cdot \mu(n) = 5$ from the expected outcome $\mu(n) = 50$ is exponential small in n. And hence, after repeating the experiment many times is X close to $\mu(n) = 50$.

$\Diamond$

2.2.1 The Chernoff-Hoeffding Bound with Limited Independence

In some cases we can't achieve fully independence among the random variables $X_1, \ldots X_n$, and hence we need a weaker result than the one given in DEFINITION 2.1. Fortunately, [2] has proved that the Chernoff-Hoeffding bound holds even if the variables are only k-wise independent. A set of random variables $X_1, \ldots, X_n$ are k-wise independent for $k \geq 1$ if every subset of size k is mutually independent, and hence $X_1, \ldots, X_n$ are fully independent if and only if it's k-wise independent for all $1 \leq k \leq n$.

DEFINITION 2.3. Let $X_1, \dots X_n$ be k-wise independent random variables for $k \geq 1$ where $0 \leq X_i \leq 1$ for all i. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \sum_{i=1}^{n} \mathbb{E}[X_i]$ where $\mathbb{E}[X_i]$ is the expected outcome of X_i. For any $0 < \epsilon \leq 1$, if $k \leq \lfloor \epsilon^2 \cdot \mu \cdot \exp\left(-\frac{1}{3}\right) \rfloor$ then:

$$\Pr[|X - \mu| \geq \epsilon \cdot \mu] \leq \exp\left(-\left\lfloor \frac{k}{2} \right\rfloor\right)$$

and if $k = \lfloor \epsilon^2 \cdot \mu \cdot \exp\left(-\frac{1}{3}\right) \rfloor$ then:

$$\Pr[|X - \mu| \geq \epsilon \cdot \mu] \leq \exp\left(-\left\lfloor \frac{\epsilon^2 \cdot \mu}{3} \right\rfloor\right)$$

where $\exp(x) = e^x$.

Remark. Notice that if the independence k is small enough depends the probability only on k (and not ϵ and μ), and otherwise is it equal the upper tail Chernoff-Hoeffding bound.

△

2.3 Lattices

Let $B = \{\vec{b_1}, \dots, \vec{b_n}\}$ be an n-linearly independent basis. A lattice $\hat{v}$ is defined by the set of vectors:

$$\hat{v} = \left\{ \sum_{i=1}^{n} x_i \cdot \vec{b_i} \mid x_i \in \mathbb{Z} \right\} = \{\vec{v_1}, \dots, \vec{v_m}\}$$

where m is the length of $\hat{v}$ and B is called the basis of $\hat{v}$.

For lattice-based constructions in cryptography to be any useful we need to base the security on some hard lattice problem. One such problem is the shortest vector problem (SVP) where we are given a lattice $\hat{v}$ and asked to find the shortest vector $\vec{v}$ in $\hat{v}$. By shortest we mean the vector $\vec{v} \in \hat{v}$ that has the smallest infinity norm $\|\vec{v}\|_\infty$ in $\hat{v}$. Sometime is the approximated version of the shortest vector problem (SVP_γ) used instead: Given a lattice $\hat{v}$ we are asked to find a vector $\vec{v} \in \hat{v}$ such that it's at most γ times larger than the shortest vector in $\hat{v}$. Notice that $\text{SVP} = \text{SVP}_1$. Furthermore, as argued in [10] is SVP_γ the central hard lattice problem because all other lattice problems can be reduced to SVP_γ. And hence, in this thesis we are only interested in the hardness of solving the shortest vector problem.

Let $L = \mathbb{Z}[X] / \langle x^n + 1 \rangle$ be a ring of polynomials. We remember that an ideal I of L is a subgroup of $(L, +)$ where $\lambda \cdot x \in I$ for all $\lambda \in L$ and $x \in I$. An ideal lattice in L is a lattice $\hat{v}$ with the extra property that for every vector $(v_0, \dots, v_{n-2}, v_{n-1})$ in $\hat{v}$ is the rotated vector with the first coordinate negated $(-v_{n-1}, v_0, \dots, v_{n-2})$ also in $\hat{v}$. Now, if we treat vectors as polynomials, i.e. $(v_0, \dots, v_{n-2}, v_{n-1})$ as $v_0 + \dots + v_{n-2} \cdot x^{n-2} + v_{n-1} \cdot x^{n-1}$, corresponds ideal lattices to ideals in L because for all $\lambda = v_0 + \dots + v_{n-2} \cdot$

$x^{n-2} + v_{n-1} \cdot x^{n-1} \in L$ is:

$$\begin{aligned}\lambda \cdot x &= (v_0 + \cdots + v_{n-2} \cdot x^{n-2} + v_{n-1} \cdot x^{n-1}) \cdot x \\ &= v_0 \cdot x + \cdots + v_{n-2} \cdot x^{n-1} + v_{n-1} \cdot x^n \\ &= v_0 \cdot x + \cdots + v_{n-2} \cdot x^{n-1} + v_{n-1} \cdot (-1) \\ &= -v_{n-1} + v_0 \cdot x + \cdots + v_{n-2} \cdot x^{n-1} \in I\end{aligned}$$

where we used the fact that we compute modulo $x^n + 1$ in L, which means that $x^n + 1 = 0$ and hence $x^n = -1$.

Ideal lattices are useful because we can build efficient collision-resistant hash functions based on the hardness of finding shortest vectors in such lattices. The hash function is defined as follows where we use the ring L from above: For any integer m and $D \subseteq L$, the family of hash functions $\mathsf{H}(L, D, m)$ mapping D^m to L is defined as:

$$\mathsf{H}(L, D, m) = \{f_{\hat{c}} \mid \hat{c} \in L^m\}$$

Now, for any $\hat{d} \in D^m$ we have that:

$$f_{\hat{c}}(\hat{d}) = \hat{c} \cdot \hat{d} = \vec{c_1} \cdot \vec{d_1} + \cdots + \vec{c_m} \cdot \vec{d_m}$$

A proof that the above hash function is collision-resistant given that the shortest vector problem is hard is given in [5].

As argued in [7] allow lattice-based protocols much less algebraic structure compared to number-theoretic protocols. E.g. the domain of a number-theoretic hash function is the whole ring while the domain of a lattice-based hash function is just a subset of the ring, which is neither closed under addition nor multiplication. This is related to the fact that factoring and discrete logarithm problems can be reduced to finding an element in the domain of some additive homomorphic function (a preimage), while the shortest vector problem can be reduced to finding a *small* element in the domain of some additive homomorphic function (a *small* preimage).

2.4 Encoding Schemes

Assume two parties, say $\mathcal{A}$ and $\mathcal{B}$, wish to communicate digitally. All they have is an unreliable channel that may result in errors in the transmitted messages. We say that the channel is noisy because it may flip or delete bits sent across. The problem here is for $\mathcal{A}$ to send a message through the channel in such a way that $\mathcal{B}$ can recover it even if some of the bits are flipped or deleted. The solution is for $\mathcal{A}$ to use an encoding scheme, which add redundancy to the message such that $\mathcal{B}$ can decode the result into $\mathcal{A}$'s original message. In fact, encoding schemes are not just used for digital communications, but also in hard drives and physical media such as CD's, which may suffer from e.g. scratches.

Let $\Sigma = \mathbb{F}_q$ be an alphabet of size $|\Sigma| = q$, i.e. the alphabet is associated with a finite field of size q. A linear code $\mathsf{C} = [n, k, d]_q$ with block length n (or codeword length), dimension k (or message length) and minimum distance

d is a linear subspace of Σ^n. Remember that $\mathsf{C} \subseteq \Sigma^n$ is a linear subspace if for every $c, c' \in \mathsf{C}$ and $\alpha \in \Sigma$ where $c = (c_1, \ldots, c_n)$ and $c' = (c'_1, \ldots, c'_n)$, it holds that:

1. C is closed under vector addition, i.e. $c + c' = (c_1 + c'_1, \ldots, c_n + c'_n) \in \mathsf{C}$.
2. C is closed under scalar multiplication, i.e. $\alpha \cdot c = (\alpha \cdot c_1, \ldots, \alpha \cdot c_n) \in \mathsf{C}$.

The Hamming distance between two codewords $c, c' \in \mathsf{C}$, denoted $\Delta(c, c')$, is the number of coordinates in which they differs:

$$\Delta(c, c') = \left|\{i | c_i \neq c'_i\}\right|$$

and the minimum distance d of C is defined as the minimum Hamming distance between all codewords in C:

$$d = \min_{c, c' \in \mathsf{C}, c \neq c'} \Delta(c, c')$$

EXAMPLE 2.4. One example of a linear code $\mathsf{C} = [n, k, d]_q$ with minimum distance $d = n - k + 1$ where $n \leq q$ is the Reed-Solomon code.

Let $P_k \subseteq \mathbb{F}_q[X]$ be the set of polynomials with degree at most $k - 1$, i.e. for all $f \in P_k$ is $\deg(f) \leq k - 1$. A message $m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}_q^k$ is encoded into the codeword $c \in \mathbb{F}_q^n$ as follows:

1. Convert the message m into the polynomial $f(x) = m_0 + m_1 \cdot x^1 + \cdots + m_{k-1} \cdot x^{k-1} \in P_k$.
2. Choose n distinct elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ (this is why we require that $n \leq q$).
3. The encoding of m is defined as the codeword $c = (f(\alpha_1), \ldots, f(\alpha_n))$.

$\Diamond$

2.4.1 Secret Sharing using Encoding Schemes

A secret sharing scheme with t-privacy and r-reconstruction is used to share a secret s between n parties where s is divided into n shares $(s_1, \ldots, s_n)$ and share s_i is given to party P_i. The secret is shared such that any set of r parties can reconstruct the secret and any set of t parties have no information about the secret. If $r = t + 1$ is the scheme called a t-threshold secret sharing scheme. Furthermore, a secret sharing scheme is said to be linear, if for any two secrets s and s' with respective shares $(s_1, \ldots, s_n)$ and $(s'_1, \ldots, s'_n)$, it holds that $(s_1 + s'_1, \ldots, s_n + s'_n)$ and $(\alpha \cdot s_1, \ldots, \alpha \cdot s_n)$ are correct shares of the secrets $s + s'$ and $\alpha \cdot s$ respectively where α is some constant.

As proved in [6] can we build a linear secret sharing scheme from the code $\mathsf{C} = [n + \ell, k, d]_q$ over the alphabet $\mathbb{F}_q$ (i.e. a finite field of size q) with $(d^\perp - \ell - 1)$-privacy and $(n + \ell - d + 1)$-reconstruction where $0 < \ell < d^\perp$. $d^\perp$ is the minimum distance of the dual code $\mathsf{C}^\perp = [n + \ell, (n + \ell) - k, d^\perp]_q$, which is defined as:

$$\mathsf{C}^\perp = \{v | \langle v, c \rangle = 0, \forall c \in \mathsf{C}\}$$

where $\langle v, c \rangle = \Sigma_{i=1}^{n+\ell} v_i \cdot c_i$ is the inner product of v and c.

To secret share $s \in \mathbb{F}_q^\ell$ we choose a codeword $c = (c_1, \ldots, c_\ell, , c_{\ell+1} \ldots, c_{\ell+n}) \in$ C uniformly at random from the q^k possible codewords such that $s = (c_1, \ldots, c_\ell)$ and where $(c_{\ell+1}, \ldots, c_{\ell+n}) \in \mathbb{F}_q^n$ are defined to be the shares of s.

2.5 Negligibility

Let $\mathcal{A}$ be an adversary whose computational power is bounded to run in time polynomial to k where k is the security parameter of the protocol Π. When defining security of Π against $\mathcal{A}$ we need a notion of the allowed success rate of $\mathcal{A}$ for Π to be secure. Assume that the only attack against Π is a brute-force attack. Since $\mathcal{A}$ is polynomial time bounded he can't complete a brute-force attack, but instead guess $\mathsf{poly}(k)$ number of random values where poly is some polynomial and hope that he guess the right one. Therefore, the probability of $\mathcal{A}$'s success $\epsilon(k)$ has to be strictly smaller than $\frac{1}{\mathsf{poly}(k)}$ for any polynomial poly. We say that the probability $\epsilon(k)$ has to be negligible in k for Π to be secure.

DEFINITION 2.5. The function $\epsilon(k)$ is negligible in k if and only if there for any polynomial poly exists a $k_0 \in \mathbb{N}$ such that $\forall k > k_0 : \epsilon(k) < \frac{1}{\mathsf{poly}(k)}$. This is also denoted as $\epsilon < \mathsf{negl}(k)$.

EXAMPLE 2.6. An example of a negligible function is $\epsilon(k) = 2^{-k}$ since it's exponential small in k.

$\Diamond$

2.6 Indistinguishability

The statistical distance between two probability distributions P and Q is a measure of how far apart they are from each other.

DEFINITION 2.7. Let P and Q be two probability distributions. Define $P(y)$ and $Q(y)$ as the probability that P and Q respectively assigns y. The statistical distance between P and Q is defined as:

$$SD(P, Q) = \sum_y |P(y) - Q(y)|$$

EXAMPLE 2.8. Consider a coin toss with $y \in \{\text{head}, \text{tail}\}$. Assume that $P(\text{head}) = P(\text{tail}) = \frac{1}{2}$, $Q(\text{head}) = \frac{1}{3}$ and $Q(\text{tail}) = \frac{2}{3}$, i.e. P corresponds to the probability distributions of a honest coin and Q of a biased coin. The statistical distance between P and Q is:

$$\begin{aligned} SD(P, Q) &= \sum_y |P(y) - Q(y)| \\ &= |P(\text{head}) - Q(\text{head})| + |P(\text{tail}) - Q(\text{tail})| \\ &= \left|\frac{1}{2} - \frac{1}{3}\right| + \left|\frac{1}{2} - \frac{2}{3}\right| = \frac{1}{6} + \frac{1}{6} = \frac{2}{6} \end{aligned}$$

$\Diamond$

A useful concept in cryptography is called indistinguishability. Let U and V be two probabilistic algorithms and D a probabilistic polynomial time distinguisher. Assume that we run both U and V on the same input x and then choose one of the output, which we denote by y. Furthermore, the length of y should be polynomial in the length of x. Now, if we input x and y to D, it tries to guess whether y is the output of U or V. And if D has a hard time, we say that U and V are indistinguishable.

DEFINITION 2.9. Let U and V be two probabilistic algorithms where U_x and V_x are probability distributions of U and V respectively on input x. Furthermore, let D be a probabilistic polynomial time distinguisher.

1. U and V are perfectly indistinguishable, written $U \sim^p V$, if and only if $SD(U_x, V_x) = 0$.
2. U and V are statistically indistinguishable, written $U \sim^s V$, if and only if $SD(U_x, V_x)$ is negligible in the length of x.
3. U and V are computationally indistinguishable, written $U \sim^c V$, if and only if $\mathsf{D}(U) \sim^s \mathsf{D}(V)$ for every D. $\mathsf{D}(U)$ is defined as the algorithm that first runs U on input x to get the output y and then runs D on input x and y. $\mathsf{D}(V)$ is defined similar for V.

Remark. Perfectly indistinguishable means that no matter how much computing power D has, it has no chance of distinguishing at all. For statistically indistinguishable D may have a slightly advantage over a random guess, but it remains negligible. Finally computationally indistinguishable means that if D has a lot of computing power, it may be able to distinguishing.

△

EXAMPLE 2.10. If we think of x as a public key for an encryption scheme, U as the encryption of the message m_1 and V as the encryption of the message m_2, then indistinguishability of U and V means that an adversary has a hard time guessing whether a ciphertext is an encryption of m_1 or m_2.

◊

2.7 Commitment Schemes

A commitment scheme is used in many modern cryptography protocols and let one part, say P, commit to a value such that he can't change it later on and it's only revealed to another party, say V, when he chooses.

DEFINITION 2.11. Let P and V be two parties and KGen a probabilistic polynomial time key generator, which on input 1^k with security parameter k outputs a public key pk. Furthermore, let $\mathsf{commit}_{\mathsf{pk}} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^*$ be a commitment scheme for pk where n and s are polynomial in k.

1. Setup: P or V runs $\mathsf{KGen}(1^k)$, which outputs the public key pk, and sends it to the other party who verifies it.

2. Commitment: P commits to the value $m \in \{0,1\}^n$ by first choosing $r \in \{0,1\}^s$ uniformly at random and then computing $c = \mathsf{commit}_{\mathsf{pk}}(m, r)$, which he sends to V.

3. Opening: P opens the value of c by revealing m and r to V, who verifies that $c = \mathsf{commit}_{\mathsf{pk}}(m, r)$.

The property that P can't change the committed value after he has sent it to V is called the *binding property* and that V doesn't know the committed value until P reveals it is called the *hiding property*. Hence, a commitment scheme comes in two flavors:

1. Unconditional binding and computational hiding: P runs $\mathsf{KGen}(1^k)$ to generate the public key pk, which he sends to V who either accepts or rejects it.

   Unconditional binding: If pk is correctly generated, then for any c exists there at most one value m such that for some r it holds that $c = \mathsf{commit}_{\mathsf{pk}}(m, r)$. In other words, even if P has infinite computing power he can't change the committed value. Furthermore, V accepts only an incorrect generated pk with probability negligible in k.

   Computational hiding: For two commitments $c = \mathsf{commit}_{\mathsf{pk}}(m, r)$ and $c' = \mathsf{commit}_{\mathsf{pk}}(m', r')$ where $m \neq m'$ and $r \neq r'$, we require that (pk, c) and (pk, c') are computational indistinguishable, i.e. $(\mathsf{pk}, c) \sim^c (\mathsf{pk}, c')$. Hence, V has a hard time guessing what is inside a commitment.

2. Computational binding and unconditional hiding: V runs $\mathsf{KGen}(1^k)$ to generate the public key pk, which he sends to P who either accepts or rejects it.

   Computational binding: Let P^* be a malicious prover who is polynomial time bounded and tries to change the committed value. We require that the probability that P^* on input pk outputs a commitment c and two valid openings (r, m) and (r', m') where $m \neq m'$ and $r \neq r'$, i.e. $\mathsf{commit}_{\mathsf{pk}}(m, r) = c = \mathsf{commit}_{\mathsf{pk}}(m', r')$, is negligible in k.

   Unconditional hiding: The commitment c reveals almost no information about the committed value m, i.e. for two commitments $c = \mathsf{commit}_{\mathsf{pk}}(m, r)$ and $c' = \mathsf{commit}_{\mathsf{pk}}(m', r')$ where $m \neq m'$ and $r \neq r'$, we require that $(\mathsf{pk}, c) \sim^s (\mathsf{pk}, c')$. Furthermore, P accepts only an incorrect generated pk with probability negligible in k. If the commitment scheme is *perfect hiding* we have that $(\mathsf{pk}, c) \sim^p (\mathsf{pk}, c')$ and P never accepts an incorrect generated pk.

Remark. It's clear that an unconditional guarantee is better that a computational one, so why do a commitment scheme not come in an unconditional binding and unconditional hiding flavor? Assume that commit is such a commitment scheme. Furthermore, assume that P has computed the commitment $c = \mathsf{commit}_{\mathsf{pk}}(m, r)$ to the value m by using the randomness r.

Since commit satisfies unconditional hiding, there must exist an $r' \neq r$ such that $c = \mathsf{commit}_{\mathsf{pk}}(m', r')$ for $m' \neq m$. If not, V could try to compute the commitment of every pair of value and randomness and compare them to c, and hereby find the committed value m. However, if P has infinite computing power he could use the same technique and find the pair $r' \neq r$ and $m' \neq m$ such that $c = \mathsf{commit}_{\mathsf{pk}}(m', r')$, and hence break the unconditional binding property.

△

EXAMPLE 2.12. The following is an example of a computational binding and unconditional hiding *bit* commitment scheme.

1. Setup: V runs KGen on input 1^k, which outputs the public key $\mathsf{pk} = (n, q, y)$ where n is a k-bit RSA modulus, q is a prime such that $q > n$ and $y = f(x)$ for $x \in_R \mathbb{Z}_n^*$. The function f is defined as $f(a) = a^q \bmod n$. Finally P verifies that $\gcd(y, n) = 1$ (i.e. pk is correctly generated).
2. Commitment: P commits to the bit b by computing $c = \mathsf{commit}_{\mathsf{pk}}(b, r) = y^b \cdot f(r) \bmod n$ where $r \in_R \mathbb{Z}_n^*$, which he sends to V.
3. Opening: P opens the value of c by revealing b and r to V, who verifies that $c = \mathsf{commit}_{\mathsf{pk}}(b, r)$.

◊

2.8 Interactive Proof Systems

Let P and V, called the prover and verifier respectively, be two interactive Turing machines with a common communication tape that allows them to send and receive messages from and to each other. Assume that P has infinite computing power and V is polynomial time bounded. In an interactive proof system (P, V) is P and V given x as common input string where P claims that the statement $x \in L$ is true for the binary language L. After running (P, V) outputs V either accept or reject, which is also denoted as $(\mathsf{P}, \mathsf{V})(x) \rightarrow$ accept/reject.

DEFINITION 2.13. Let (P, V) be an interactive proof system for the binary language $L \subset \{0,1\}^*$ where P claims that the statement $x \in L$ is true. An interactive proof system satisfies the following two properties:

1. Completeness: If P and V follows the protocol on input x where $x \in L$, then the probability that V outputs reject is negligible in the length of x.
2. Soundness: For any malicious prover P^*, if P^* and V follows the protocol on input x where $x \notin L$, then the probability that V outputs accept is negligible in the length of x.

Remark. A more realistic version of the above protocol is called an *interactive argument* where both P and V are polynomial time bounded. In this version has P some extra auxiliary input δ, which allows him to convince V to output accept.

△

2.8.1 Proof of Knowledge Systems

Let $R \subset \{0,1\}^* \times \{0,1\}^*$ be a binary relation. We say that $(w, x) \in R$ if w is a solution to x, which is an instance of some computational problem. In other words, w is a *witness* for x.

EXAMPLE 2.14. The following is an example of a relation R that contains the discrete logarithm problems and their solutions:

$$R = \{(w, x) \mid x = (p, q, g, h), \mathrm{ord}(g) = \mathrm{ord}(h) = q, h = g^w \bmod p\}$$

where p and q are primes.

$\Diamond$

A variant of an interactive proof system is called a *proof of knowledge system* where the prover P now claims that he "know" some piece of information w that enable him to prove that the statement $x \in L_R$ is true (such as a secret key corresponding to a public key). L_R is defined as the language of problems x for which there exists witnesses w such that $(w, x) \in R$ for the relation R:

$$L_R = \{x \mid \exists w, (w, x) \in R\}$$

Therefore, after running P and V on common input $x \in L_R$ where P is also given w as private input, which he claims to know the value of such that $(w, x) \in R$, V outputs either accept or reject. This is also denoted as $(\mathsf{P}(w), \mathsf{V})(x) \rightarrow$ accept/reject.

Since it's difficult to define that a machine "know" some piece of information, we say that it know some piece of information if it can be used to compute the relevant information efficiently. Hence, we let a knowledge extractor Ext, which is only given public information, interact with P and if Ext can extract a correct witness from P to the problem x in the relation R, then P must know the value of w such that $(w, x) \in R$.

DEFINITION 2.15. The protocol (P, V) is a proof of knowledge for the relation R with knowledge error $\kappa(x) \in [0; 1]$ if it satisfies the following two properties:

1. Knowledge completeness: If P and V follows the protocol on input x and private input w to P where $(w, x) \in R$, then V always accept.
2. Knowledge soundness: Let Ext be a probabilistic knowledge extractor, which gets input x and rewindable black-box access to P^*. Furthermore, let $p(x)$ be the probability that any P^* convinces V to output accept. We require that the following holds: For any P^* there exists a polynomial poly such that whenever $p(x) > \kappa(x)$ then Ext outputs a correct witness to x in expected time at most $\frac{1}{\mathsf{poly}(p(x)-\kappa(x))}$.

Remark 1. We can think of the knowledge error $\kappa(x)$ as the probability that a malicious prover P^* can convince V to output accept without knowing a correct witness to x.

$\triangle$

Remark 2. The running time of Ext is related to the probability that P^* convinces V to output accept because the better P^* is to convince V, the more likely it's that P^* know the value of a correct witness. Therefore, the probability that Ext extracts a correct witness is at least $\mathsf{poly}(p(x) - \kappa(x))$, and hence the expected running time of Ext is at most $\frac{1}{\mathsf{poly}(p(x)-\kappa(x))}$.

△

2.9 Zero-Knowledge Protocols

Define P as the set of problems whose solutions can be found in polynomial time and NP as the set of problems whose solutions can be verified in polynomial time. One of the big question in computer science is whether $\mathsf{P} = \mathsf{NP}$ or not, i.e. if the solution to a problem can be verified in polynomial time, can it be found in polynomial time? Problems that are at least as hard as the hardest problem in NP are said to be NP-hard problems, i.e. if H is a NP-hard problem then every problem L in NP can be reduced to H in polynomial time. Furthermore, if H is also a NP problem (i.e. solutions to H can be verified in polynomial time), we say that H is a NP-complete problem.

EXAMPLE 2.16. An example of a NP problem is the set of RSA prime factors. Given a prime factor $n = p \cdot q$ where p and q are primes, it's hard to find p and q, but given p and q it's easy to compute n (i.e. it's easy to verify the solution).

◊

Another useful technique in cryptography is called *the simulator paradigm*: Let X and Y be two pieces of information. We want to argue that some party in a protocol doesn't learn anything else than X after seeing Y. The simulator paradigm says that if Y can efficiently be computed from X, then a party learns no more than X after seeing Y.

EXAMPLE 2.17. If we think of an interactive proof system (P, V) with P as the prover and V as the verifier, X could correspond to the fact that P's claim is true and Y to all the messages sent during the protocol between P and V. Therefore, if we efficiently can simulate the protocol without interacting with P (i.e. efficiently compute Y from X), V learns only that P's claim is true and nothing else.

◊

Using the concept illustrated in EXAMPLE 2.17 can we now define the notion of a zero-knowledge protocol: A zero-knowledge protocol is an interactive proof system or an interactive argument with an extra zero-knowledge property, which says that even a malicious verifier V^* learns nothing else except that $x \in L$ (i.e. V^* learns only one bit of information).

DEFINITION 2.18. Let (P, V) be an interactive proof system or interactive argument for the language L with P as the prover and V as the verifier. (P, V) is computational zero-knowledge if for any probabilistic polynomial time bounded malicious verifier V^*, there exists a

simulator $\mathsf{Sim}_{\mathsf{V}^*}$, which runs in expected polynomial time, such that $\mathsf{Sim}_{\mathsf{V}^*}(x) \sim^c (\mathsf{P}, \mathsf{V}^*(\delta))(x)$ where P claims that $x \in L$ and δ is some auxiliary input to V^*.

Remark 1. We can for some protocols (P, V) achieve perfect zero-knowledge if $\mathsf{Sim}_{\mathsf{V}^*}(x) \sim^p (\mathsf{P}, \mathsf{V}^*(\delta))(x)$ or statistical zero-knowledge if $\mathsf{Sim}_{\mathsf{V}^*}(x) \sim^s (\mathsf{P}, \mathsf{V}^*(\delta))(x)$. We have that perfect zero-knowledge implies statistical zero-knowledge, which again implies computational zero-knowledge.

△

Remark 2. The auxiliary input δ, which is only given to the malicious verifier V^*, corresponds to previous information that V^* has gained from earlier executions of the protocol. V^* may use δ to trick the prover P into revealing more information.

△

Remark 3. The reason why a prover with infinite computing power can't cheat the verifier, but the simulator with no special knowledge can, is because the simulator is allowed to generate the messages in the protocol in an arbitrary order while the prover is forced by the verifier to follow the protocol in the correct order.

△

A variant of the simulator is called a *perfect honest-verifier simulator*, which is denoted by $\mathsf{Sim}_{\mathsf{V}}$. A simulator is a perfect honest-verifier simulator if it holds that $\mathsf{Sim}_{\mathsf{V}}(x) \sim^p (\mathsf{P}, \mathsf{V})(x)$ and it's polynomial time bounded, i.e. in polynomial time can it simulate the protocol between a honest prover and a honest verifier.

LEMMA 2.19 (The Rewinding Lemma)**.** *Let* (P, V) *be an interactive proof system or interactive argument for the language* L *with* P *as the prover and* V *as the verifier. Furthermore, let* $\mathsf{Sim}_{\mathsf{V}}$ *be a perfect honest-verifier simulator for* (P, V)*, i.e.* $\mathsf{Sim}_{\mathsf{V}}(x) \sim^p (\mathsf{P}, \mathsf{V})(x)$*. Assume that the conversation between* P *and* V *has the form of* (a, b, z) *where* P *first sends* a*,* V *responds with a random bit* b *and* P *replies with* z*.* (P, V) *is then a perfect zero-knowledge protocol.*

Remark. If the simulator $\mathsf{Sim}_{\mathsf{V}}$ is a statistical or computational honest-verifier simulator for (P, V), i.e. $\mathsf{Sim}_{\mathsf{V}}(x) \sim^s (\mathsf{P}, \mathsf{V})(x)$ or $\mathsf{Sim}_{\mathsf{V}}(x) \sim^c (\mathsf{P}, \mathsf{V})(x)$, then is (P, V) a statistical or computational zero-knowledge protocol.

△

It has been proved that we can build a zero-knowledge protocol from the NP-complete Circuit Satisfiability (Circuit SAT) problem[1], which implies that any NP problem can be turned into a zero-knowledge protocol: Given a problem instance $x \in L$ where $L \in \mathsf{NP}$ we first construct from x a Boolean circuit, which is only satisfiable when $x \in L$, and then use the zero-knowledge protocol for Circuit SAT. The zero-knowledge protocol for Circuit SAT is however inefficient, because we need at least 10,000 or 100,000 binary gates for interesting problems as argued in [11]. Fortunately, we can for some NP problems construct ad-hoc protocols such as the one in EXAMPLE 2.20,

[1]The Circuit SAT problem is the decision problem of determining whether a given Boolean circuit has an assigment of its inputs that makes the output true.

which is based on the graph isomorphism problem. But in e.g. identification scenarios where the verifier is allowed to learn some information, just not enough such that he can impersonate the prover, we can build very efficient protocols called sigma protocols, which satisfies a weaker property called special honest-verifier zero-knowledge.

EXAMPLE 2.20. The following protocol is an example of an ad-hoc perfect zero-knowledge protocol for the NP graph isomorphism problem. The prover P and verifier V are given two graphs G_0 and G_1 as input where both has n nodes. P then claims that G_0 and G_1 are isomorphic, i.e. there exists a permutation π such that $\pi(G_0) = G_1$. The protocol repeats the following steps n times:

1. P chooses a permutation ϕ on n points uniformly at random and sends $H = \phi(G_0)$ to V.
2. V chooses a challenge bit b uniformly at random and sends b to P.
3. If $b = 0$ sends P the permutation $\psi = \phi^{-1}$ to V, otherwise sends he $\psi = \pi \cdot \phi^{-1}$ to V. Finally V outputs accept if and only if $\psi(H) = G_b$.

As argued is the protocol perfect zero-knowledge, and hence there exists a perfect honest-verifier simulator $\mathsf{Sim_V}$ such that $\mathsf{Sim_V}(x) \sim^p (\mathsf{P}, \mathsf{V})(x)$. $\mathsf{Sim_V}$ repeats the following steps n times:

1. $\mathsf{Sim_V}$ chooses a bit c and permutation ψ uniformly at random and sends $H = \psi^{-1}(G_c)$ to V.
2. $\mathsf{Sim_V}$ receives the challenge bit b from V. If $b = c$ outputs $\mathsf{Sim_V}$ the conversation (H, b, ψ) (i.e. the simulator has completed one iteration), otherwise resets $\mathsf{Sim_V}$ the verifier and goes back to step (1).

$\Diamond$

2.10 Sigma Protocols

Let P, the prover, and V, the verifier, be polynomial time bounded and R some binary relation. A sigma protocol (or Σ-protocol) (P, V) for the relation R is a 3-step protocol where both P and V are given $x \in L_R$ as common input and P is given w as private input. P then tries to convince V about that he know the value of the witness w to the problem x in the relation R, i.e. $(w, x) \in R$, without revealing enough information that makes V able of executing the protocol as the prover. After running the protocol V outputs either accept or reject, which is also denoted as $(\mathsf{P}(w), \mathsf{V})(x) \rightarrow$ accept/reject.

DEFINITION 2.21. A sigma protocol (P, V) for the relation R is a 3-step protocol of the form (a, e, z) where P first sends a, V responds with a random t-bit challenge e and P replies with z. Furthermore, the following three properties should hold:

1. Completeness: If P and V follows the protocol on input x and private input w to P where $(w, x) \in R$, then V always accept.

2. Special soundness: Given two accepting conversations (a, e, z) and (a, e', z') for the same x where $e \neq e'$, there exists a probabilistic polynomial time knowledge extractor Ext, which on input (x, a, e, e', z, z') can extract a correct witness w^* from P such that $(w^*, x) \in R$.

3. Special honest-verifier zero-knowledge (sHVZK): There exists a probabilistic polynomial time simulator Sim, which on input x and a random challenge e outputs an accepting conversation (a, e, z) with exactly the same probability distribution as a conversation between a honest P and V on input x. This is also denoted as $\mathsf{Sim}(x, e) \sim^p (\mathsf{P}(w), \mathsf{V})(x)$.

Remark 1. The special soundness property implies that (P, V) is an interactive proof system for the language L_R with soundness error $\sigma(x) = 2^{-t}$ where t is the length of the challenge e. Therefore, a malicious P*, who doesn't know the value of a correct witness, can convince V to output accept with probability at most 2^{-t}. Furthermore, (P, V) is also a proof of knowledge with knowledge error $\kappa(x) = 2^{-t}$.

△

Remark 2. One of the reason why a sigma protocol is such an impoartant protocol, is because it can be used as an identification scheme, a signature scheme, a zero-knowledge protocol or a commitment scheme. See [9] for a description of how the different protocols can be construed using only a sigma protocol and their proof of security.

△

Remark 3. We will later on in the thesis refer to all three properties in DEFINITION 2.21 as *perfect* completeness, *perfect* special soundness and *perfect* special honest verifier zero-knowledge to distinguish them from some slightly different properties defined in DEFINITION 2.23. The properties in DEFINITION 2.23 will later on be referred to as *statistical* completeness, *statistical* special soundness and *computational* special honest verifier zero-knowledge.

△

EXAMPLE 2.22. One example of a sigma protocol is Schnorrs sigma protocol for the following relation R with the primes p and q, which is also described in EXAMPLE 2.14:

$$R = \{(w, x) \mid x = (p, q, g, h), \mathrm{ord}(g) = \mathrm{ord}(h) = q, h = g^w \bmod p\}$$

In Schnorrs sigma protocol tries the prover P to convince the verifier V about that he knows the value of the discrete logarithm w to h:

1. P chooses r uniformly at random in $\mathbb{Z}_q$ and sends $a = g^r \bmod p$ to V.

2. V chooses a t-bit challenge e uniformly at random where $2^t \leq q$ and sends e to P.

3. P sends $z = r + e \cdot w \bmod q$ to V, who outputs accept if and only if $g^z \equiv a \cdot h^e \pmod p$, p and q are primes and $\mathrm{ord}(g) = \mathrm{ord}(h) = q$.

◊

2.10.1 Statistically Secure Sigma Protocols

The protocol we present in Section 4.2 can't meet the three properties for a sigma protocol as defined in DEFINITION 2.21. Instead we define a new type of protocol that we call a *statistically secure sigma protocol*, which satisfies some slightly weaker properties called *statistical* completeness, *statistical* special soundness and *computational* special honest verifier zero-knowledge.

DEFINITION 2.23. A statistically secure sigma protocol (P, V) for the relation R is a 3-step protocol of the form (a, e, z) where the following three properties should hold:

1. Statistical completeness: If P and V follows the protocol on input x and private input w to P where $(w, x) \in R$, then the probability that V outputs reject is negligible in the length of the challenge e.
2. Statistical special soundness: Let (a, e, z) and (a', e', z') be two accepting conversations for the same x where $e \neq e'$. Furthermore, let Ext be a probabilistic polynomial time knowledge extractor. The probability that Ext on input (x, a, a', e, e', z, z') *can't* extract a correct witness from the prover is negligible in the length of x.
3. Computational special honest-verifier zero-knowledge: There exists a probabilistic polynomial time simulator Sim such that $\mathsf{Sim}(x, e) \sim^c (\mathsf{P}(w), \mathsf{V})(x)$.

2.10.2 sHVZK vs. HVZK

In the literature some authors require that a sigma protocol satisfies honest-verifier zero-knowledge (HVZK) instead of *special* honest-verifier zero-knowledge (sHVZK). The difference is that the HVZK simulator is allowed to choose the challenge while the sHVZK simulator is given the challenge as input. As argued in [8] is sHVZK a stronger property than HVZK because sHVZK implies HVZK while the opposite is not generally true. However, from a sigma protocol that satisfies HVZK can we construct a slightly less efficient sigma protocol that satisfies sHVZK.

Using the following construction can we build a HVZK simulator $\mathsf{Sim}^{\mathrm{HVZK}}$ from a sHVZK simulator $\mathsf{Sim}^{\mathrm{sHVZK}}$ where $\mathsf{Sim}^{\mathrm{HVZK}}$ is only given x as input:

1. $\mathsf{Sim}^{\mathrm{HVZK}}$ chooses a t-bit challenge e uniformly at random.
2. $\mathsf{Sim}^{\mathrm{HVZK}}$ runs $\mathsf{Sim}^{\mathrm{sHVZK}}(x, e)$, which outputs an accepting conversation (a, e, z) for x with exactly the same probability distribution as a conversation in the real protocol between a honest prover P and verifier V.
3. $\mathsf{Sim}^{\mathrm{HVZK}}$ outputs the conversation (a, e, z) for x.

On the other hand, if we tries to build a sHVZK simulator $\mathsf{Sim}^{\mathrm{sHVZK}}$ from a HVZK simulator $\mathsf{Sim}^{\mathrm{HVZK}}$ would we end up with a simulator whose expected running time is exponential in the length of the challenge. This is

because $\mathsf{Sim}^{\text{sHVZK}}(x, e')$ would run $\mathsf{Sim}^{\text{HVZK}}(x)$ until it outputs a conversation (a, e, z) with $e = e'$. Because the length of the challenge is t-bit, we have that the probability that $e = e'$ is $p = \frac{1}{2^t}$, and hence the expected running time is $\frac{1}{p} = 2^t$, which is exponential large in t. Hence, we can't build a polynomial time sHVZK simulator from a HVZK simulator. However, given a sigma protocol (P, V) for the relation $(w, x) \in R$ that satisfies HVZK can we build a slightly less efficient sigma protocol $(\mathsf{P}', \mathsf{V}')$ for the same relation R that satisfies sHVZK by XORing the challenge with a random t-bit string: P first computes a using some randomness r, chooses a t-bit string e' uniformly at random and sends (a, e') to V. V then chooses a t-bit challenge e uniformly at random and sends $(e \oplus e')$ to P. P then use the witness w, the randomness r and the challenge $(e \oplus e')$ to compute the response z, which he sends to V, who verifies that $(a, (e \oplus e'), z)$ is an accepting conversation for x. A proof that $(\mathsf{P}', \mathsf{V}')$ is a sigma protocol satisfing sHVZK is given in [8].

2.11 The Random Oracle Model

Let (P, V) be a sigma protocol for the relation R with P as the prover and V as the verifier. One way to improve (P, V) is to let P use a random oracle $\mathcal{O}$, which implements a truly random function, to compute the t-bit challenge e. Hence, P first computes a using some randomness r and then use $\mathcal{O}$ on input a to get the challenge e. He then use his witness, the randomness r and the challenge e to compute the response z. Finally he sends (a, z) to V, who first use $\mathcal{O}$ on input a to get the challenge e and then verifies the response. This new protocol is called a non-interactive sigma protocol and the transformation from the interactive version to the non-interactive version is often refereed to as *the Fiat-Shamir transformation* or *the Fiat-Shamir heuristic*. However, the non-interactive version is only secure in *the random oracle model* because in the real world there don't exists functions implementing truly random functions.

You may have notice that a malicious prover now has more power because he can ask the oracle a polynomial number of times (since he is polynomial time bounded) until he receives a challenge he may be able of answering. But if the number of challenges are exponential large this is not a feasible strategy. Furthermore, since the provers access to the random oracle $\mathcal{O}$ removes any influence from a malicious verifier, which is equivalent to forcing the verifier to be honest, we have that a non-interactive sigma protocol is *perfect zero-knowledge* in the random oracle model.

DEFINITION 2.24. A random oracle $\mathcal{O}$ implements a truly random function and works as follows:

1. On input x, which $\mathcal{O}$ has not seen before, it returns a truly random t-bit string y and stores (x, y) in a table for further reference.
2. On input x, which $\mathcal{O}$ has seen before, it finds (x, y) in its table and returns y.

Proving the special soundness property for a non-interactive sigma protocol in the random oracle model is different from proving it for an interactive sigma protocol in the standard model because the prover P receives the challenge from an oracle and not the verifier. Hence, P would not be able of producing two accepting conversations (a, e, z) and (a, e', z') with different challenges $e \neq e'$ because the first message a in both conversations have to be the same and the oracle would therefore always return the same challenge. Fortunately, then [4] has presented a useful lemma called *the Forking Lemma*.

The Forking Lemma basically says that, if a possible malicious prover P^* can produce within some time bound an accepting conversation with high probability, then there exists a polynomial time bounded machine $\mathcal{M}$ which can produce two accepting conversations with different challenges by using an *oracle replay attack* on P^*. In an oracle replay attack runs $\mathcal{M}$ the possible malicious prover P^* a polynomial number of times using a different oracle in each run (i.e. a different truly random function). And if $\mathcal{M}$ is able of producing two accepting conversations, then we can use the same technique as in the standard model to extract a correct witness from P^* in polynomial time.

Chapter 3

A General Framework for Protocols with Abort

3.1 The General Framework with Abort

In this section we present a general framework with abort for protocols with abort. We have two parties in the framework, a prover P and a verifier V, who both are polynomial time bounded. In the framework claims P that he know the value of the witness $\vec{w}$ to the problem x in the relation R, i.e. $(\vec{w}, x) \in R$. P sends first a commitment to V, who responds with a challenge. P then computes a reply and if this reply is outside some interval he chooses to abort the framework. Since we don't define what should happen when P chooses to abort, the framework should not be used as a standalone protocol in its current form. We can therefore think of the framework as a building block from which we can construct a statistically secure sigma protocol (as we have done in Section 4.2).

The setup of the framework is as follows: Let $f : \mathbb{Z}^n \to (G, \circ)$ be a function mapping vectors from $\mathbb{Z}^n$ of length $n \geq 1$ to group elements in G. Furthermore, the function has to be additive homomorphic, i.e. $f(\vec{c} + \vec{d}) = f(\vec{c}) \circ f(\vec{d})$ for all $\vec{c}, \vec{d} \in \mathbb{Z}^n$. The relation $(\vec{w}, x) \in R$ is defined as $\vec{w} \in \mathbb{Z}^n$ such that $\|\vec{w}\|_\infty \leq B$ for the bound $B \geq 1$ and $x = (f, y)$ where $y = f(\vec{w})$. The prover P and verifier V are both given the problem x as common input and P is given the witness $\vec{w}$ as private input such that $(\vec{w}, x) \in R$ for the relation R. Furthermore, let commit be a commitment scheme with public key pk, which is either unconditional binding and computational hiding or computational binding and perfect hiding (see Section 2.7 for the definition of the two flavors). If commit is an unconditional binding and computational hiding commitment scheme can we achieve *perfect* special soundness and *computational* special honest-verifier zero-knowledge. Otherwise, if it's a computational binding and perfect hiding commitment scheme can we achieve *statistical* special soundness and *perfect* special honest-verifier zero-knowledge. See TABLE 3.1 for a comparison of the framework using the two different flavors of the commitment scheme and DEFINITION 2.21 and DEFINITION 2.23 for the difference between the properties.

The commitment scheme is used to hide the value of the first message in the framework if P chooses to abort, which he do if the response to the challenge is outside of the interval $I = [-(S \cdot B - B); S \cdot B - B]$. The reason why this is important is shown in the proof of THEOREM 3.1 and by comparing this proof with the proof of THEOREM 3.2 is it shown why

TABLE 3.1: Comparison of the special soundness and special honest-verifier zero-knowledge (sHVZK) properties for the general framework with abort using the two different flavors of the commitment scheme. Let $\mathsf{commit}^{ub,ch}$ be an unconditional binding and computational hiding commitment scheme and $\mathsf{commit}^{cb,ph}$ a computational binding and perfect hiding commitment scheme.

	$\mathsf{commit}^{ub,ch}$	$\mathsf{commit}^{cb,ph}$
Special soundness	Perfect	Statistical
sHVZK	Computational	Perfect

the interval I is important when we try to achieve perfect special honest-verifier zero-knowledge. The general framework with abort is defined as follows:

PROTOCOL 3.1: The general framework with abort

1. The prover P chooses $\vec{r}$ uniformly at random from $\mathbb{Z}^n$ such that $\|\vec{r}\|_\infty \leq S \cdot B$ for $S \geq 1$, computes $a = f(\vec{r})$ and sends the commitment $com = \mathsf{commit}_{\mathsf{pk}}(a, s)$ to the verifier V. $s \in_R \mathbb{Z}$ is the randomness used in the commitment scheme and S defines how large an interval $\vec{r}$ is chosen from compared to $\vec{w}$.

2. V chooses a challenge bit b uniformly at random and sends b to P.

3. P computes $\vec{z} = \vec{r} + b \cdot \vec{w}$. If $\vec{z} \in I^n$ for the interval $I = [-(S \cdot B - B); S \cdot B - B]$ he sends $\mathcal{Z} = (\vec{z}, a, s)$ to V (i.e. he sends the response and opens the commitment), otherwise he sends $\mathcal{Z} = \perp$ to V (i.e. he aborts the protocol). Finally, if $\mathcal{Z} \neq \perp$ then V verifies the response by checking that $com = \mathsf{commit}_{\mathsf{pk}}(a, s)$ and $f(\vec{z}) = a \circ y^b$, i.e. $(com, b, \mathcal{Z})$ is an accepting conversation for x.

Illustration of the protocol

PROTOCOL 3.1: The general framework with abort (continued)

Prover $P(\vec{w}, x)$		**Verifier** $V(x)$
$\vec{r} \in_R \mathbb{Z}^n$		
$a = f(\vec{r})$		
$s \in_R \mathbb{Z}$		
$com = \mathsf{commit}_{\mathsf{pk}}(a, s)$		
	$\xrightarrow{com}$	
	$\xleftarrow{b \in_R \{0,1\}}$	
$\vec{z} = \vec{r} + b \cdot \vec{w}$		
if $\vec{z} \in I^n$ then		
$\mathcal{Z} = (\vec{z}, a, s)$		
else $\mathcal{Z} = \perp$		
	$\xrightarrow{\mathcal{Z}}$	
		if $\mathcal{Z} \neq \perp$:
		accept iff
		$com = \mathsf{commit}_{\mathsf{pk}}(a, s)$
		and $f(\vec{z}) = a \circ y^b$

THEOREM 3.1. *Let* $\mathsf{commit}^{ub,ch}$ *be an unconditional binding and computational hiding commitment scheme and* $\mathsf{commit}^{cb,ph}$ *a computational binding and perfect hiding commitment scheme.*

The general framework with abort presented in this section satisfies completeness, but where the prover P *chooses to abort with probability* $\Pr[\vec{z} \notin I^n]$ *(see Equation (3.1)), special soundness (perfect if* $\mathsf{commit}^{ub,ch}$ *is used and statistical if* $\mathsf{commit}^{cb,ph}$ *is used) where the extracted witness has size* $\|\vec{w}^*\|_\infty \leq 2 \cdot (S \cdot B - B)$ *and special honest-verifier zero-knowledged (computational if* $\mathsf{commit}^{ub,ch}$ *is used and perfect if* $\mathsf{commit}^{cb,ph}$ *is used).*

Proof of Theorem 3.1. The proof is given in Section 3.1.1. □

3.1.1 Proof of THEOREM 3.1

Completeness with Abort

Since the prover P may choose to abort the protocol with some probability (see Equation (3.1)), we say that the protocol satisfies a property called *completeness with abort*, which is slightly different from the completeness property defined in DEFINITION 2.21. Assume that both the prover P and the verifier V are honest and follows the protocol. Furthermore, assume that P know a correct witness $\vec{w}$ to the problem x such that $(\vec{w}, x) \in R$. The

protocol satisfies completeness with abort if V always output accept *when* $\mathcal{Z} \neq \perp$. Hence, we have to prove that a honest prover can always produce an accepting conversation $(com, b, \mathcal{Z})$ such that $com = \mathsf{commit}_{\mathsf{pk}}(a, s)$ and $f(\vec{z}) = a \circ y^b$ when $\mathcal{Z} \neq \perp$.

Assume that $\mathcal{Z} \neq \perp$. Since P is honest he can always correctly open the commitment com, and hence the first check performed by V is true. For $b = 0$ we have that:

$$\begin{aligned} f(\vec{z}) &= a \circ y^b \\ f(\vec{r} + b \cdot \vec{w}) &= a \circ y^b \\ f(\vec{r} + 0 \cdot \vec{w}) &= a \circ y^0 \\ f(\vec{r}) &= a \end{aligned}$$

which is true according to the definition of a. For $b = 1$ we have that:

$$\begin{aligned} f(\vec{z}) &= a \circ y^b \\ f(\vec{r} + b \cdot \vec{w}) &= a \circ y^b \\ f(\vec{r} + 1 \cdot \vec{w}) &= a \circ y^1 \\ f(\vec{r} + \vec{w}) &= f(\vec{r}) \circ f(\vec{w}) \end{aligned}$$

which is true because the function f is additive homomorphic.

The probability that P chooses to abort is:

$$\begin{aligned} \Pr[\vec{z} \notin I^n] &= 1 - \Pr[\vec{z} \in I^n] \\ &= 1 - \left(\frac{2 \cdot (S \cdot B - B) + 1}{2 \cdot (S \cdot B) + 1} \right)^n \end{aligned} \tag{3.1}$$

because for every $i = 1, 2, \ldots, n$ there is $2 \cdot (S \cdot B - B) + 1$ possibilities of r_i in $\vec{r} = (r_1, \ldots, r_n)$ that may lead to $\vec{z} \in I^n$ where $2 \cdot (S \cdot B) + 1$ is the size of the interval that $\vec{r}$ is chosen from. A formal proof for $\Pr[\vec{z} \in I^n]$ is given below.

The only value that P chooses during the protocol is the randomness $\vec{r}$, and hence it depends on the values of r_i in $\vec{r}$ whether P has to abort or not. Because V sends a bit as challenge we have two cases: $b = 0$ and $b = 1$. For $b = 0$ is $\vec{z} = \vec{r} + b \cdot \vec{w} = \vec{r} + 0 \cdot \vec{w} = \vec{r}$, i.e. we don't have to consider the value of $\vec{w}$. We know that $\|\vec{r}\|_\infty \leq S \cdot B$ and $\|\vec{z}\|_\infty \leq S \cdot B - B$ when $\vec{z} \in I^n$. Therefore, if we want $\vec{z} \in I^n$ then for all $i = 1, \ldots, n$ must:

$$r_i \in [-(S \cdot B - B); S \cdot B - B]$$

which is an interval of size $2 \cdot (S \cdot B - B) + 1$. And hence, for $b = 0$ is $\Pr[\vec{z} \in I^n] = \left(\frac{2 \cdot (S \cdot B - B) + 1}{2 \cdot (S \cdot B) + 1} \right)^n$.

For $b = 1$ is $\vec{z} = \vec{r} + \vec{w}$. Let $\vec{w}$ be any value such that $\|\vec{w}\|_\infty \leq B$, e.g. the one at point (a) in FIGURE 3.1. This implies that $\vec{z}$ lies between point (b) and (c) in the figure, but if we want $\vec{z} \in I^n$ then $\vec{z}$ must not be in the interval illustrated by I_- and I_+ in the figure. To ensure this, $\vec{r}$ must not be from the interval illustrated by J_- and J_+ in the figure, which are the same intervals as I_- and I_+, just shifted such that their range are inside the possible values of $\vec{r}$ (remember that $\|\vec{r}\|_\infty \leq S \cdot B$). Because of symmetry is

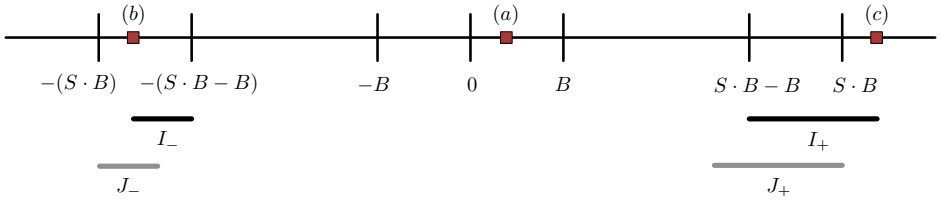

FIGURE 3.1

the size of the union of J_- and J_+ equal $|J_-| + |J_+| = 2 \cdot B$. This implies that if we want $\vec{z} \in I^n$ then for all $i = 1, \ldots, n$ must r_i be from an interval of size $2 \cdot (S \cdot B) + 1 - 2 \cdot B = 2 \cdot (S \cdot B - B) + 1$. And hence, for $b = 1$ is $\Pr[\vec{z} \in I^n] = \left(\frac{2 \cdot (S \cdot B - B) + 1}{2 \cdot (S \cdot B) + 1}\right)^n$.

(Statistical/Perfect) Special Soundness

We will first argue why the protocol satisfies *perfect* special soundness as defined in DEFINITION 2.21 when the used commitment scheme commit is unconditional binding and computational hiding. And then why it satisfies *statistical* special soundness as defined in DEFINITION 2.23 when commit is computational binding and perfect hiding.

For perfect special soundness we have to prove that if a (possible malicious) prover P^* can produce two accepting conversations $(com, b, \mathcal{Z} = (\vec{z}, a, s))$ and $(com, b', \mathcal{Z}' = (\vec{z}\,', a', s'))$ for the same x where $b \neq b'$, then there exists a probabilistic polynomial time knowledge extractor Ext, which on input $(x, a, a', b, b', \vec{z}, \vec{z}\,')$ can extract a correct witness $\vec{w}^*$ from P^* such that $(\vec{w}^*, x) \in R$. Notice that, since commit is unconditional binding we have that $a = a'$, because after P^* has produced the first accepting conversation with $com = \mathsf{commit}_{\mathsf{pk}}(a, s)$, he can't even if he has infinite computing power find a different $a' \neq a$ for the second conversation such that $com = \mathsf{commit}_{\mathsf{pk}}(a', s')$. Therefore, two accepting conversations with $b = 0$ and $b' = 1$ gives the following two equations:

$$\begin{aligned} f(\vec{z_0}) &= a \circ y^b \\ &= a \circ y^0 \\ &= a \end{aligned}$$

$$\begin{aligned} f(\vec{z_1}) &= a \circ y^{b'} \\ &= a \circ y^1 \\ &= a \circ y \end{aligned}$$

Using $f(\vec{z_0}) = a$ in $f(\vec{z_1}) = a \circ y$ and the fact that both G is a group with composition $\circ$ (i.e. every elements in G has an inverse) and $\mathbb{Z}^n$ is a group

with composition "+" (i.e. the inverse is "−") we have that:

$$\begin{aligned} f(\vec{z_1}) &= a \circ y \\ f(\vec{z_1}) &= f(\vec{z_0}) \circ y \\ f(\vec{z_1}) \circ f(\vec{z_0})^{-1} &= y \\ f(\vec{z_1} - \vec{z_0}) &= y \end{aligned}$$

Hence, the knowledge extractor Ext outputs the witness $\vec{w}^* = \vec{z_1} - \vec{z_0}$, which satisfies that $(\vec{w}^*, x) \in R$ and has size $\|\vec{w}^*\|_\infty = \|\vec{z_1} - \vec{z_0}\|_\infty \leq 2 \cdot (S \cdot B - B)$ because both $\|\vec{z_1}\|_\infty \leq S \cdot B - B$ and $\|\vec{z_0}\|_\infty \leq S \cdot B - B$.

We will now argue what happens to the above proof if the used commitment scheme is computational binding and perfect hiding. As defined in Section 2.7 (the definitions of the two flavors of a commitment scheme) is the probability that a polynomial time bounded P^* finds a different $a' \neq a$, after he has produced the first accepting conversation such that $\mathsf{commit}_{\mathsf{pk}}(a, s) = com = \mathsf{commit}_{\mathsf{pk}}(a', s')$, negligible in the security parameter. Remember that the security parameter was given as input to the key generator algorithm that generated the public key pk. Therefore, if P^* succeed in finding a different $a' \neq a$ it implies that the knowledge extractor Ext can't output a correct witness. Hence, the protocol satisfies statistical special soundness.

(Computational/Perfect) Special Honest-Verifier Zero-Knowledge

We will first argue why the protocol satisfies *perfect* special honest-verifier zero-knowledge (sHVZK) as defined in DEFINITION 2.21 when the used commitment scheme commit is computational binding and perfect hiding. And then why it satisfies *computational* sHVZK as defined in DEFINITION 2.23 when commit is unconditional binding and computational hiding.

For perfect special honest-verifier zero-knowledge we have to construct a probabilistic polynomial time simulator Sim, which on input x and random challenge bit b_{s} outputs a conversation with the same probability distribution as a conversation from the real protocol between a honest prover P and verifer V, i.e. $\mathsf{Sim}(x, b_{\mathsf{s}}) \sim^p (\mathsf{P}(\vec{w}), \mathsf{V})(x)$ where $(\vec{w}, x) \in R$. But before we prove this statement and for a better intuition about why we have to use a commitment scheme in the framework, we will first describe the framework without the commitment scheme and why we can't simulate this version of the framework correct. After that we construct the correct simulator for the framework with a commitment scheme and argue why this simulation is perfect.

Why the commitment scheme is needed. The general framework with abort would have been defined as follows if we didn't had to use a commitment scheme:

1. The prover P chooses $\vec{r}$ uniformly at random from $\mathbb{Z}^n$ such that $\|\vec{r}\|_\infty \leq S \cdot B$ for $S \geq 1$ and sends $a = f(\vec{r})$ to the verifier V.

2. V chooses a challenge bit b uniformly at random and sends b to P.

3. P computes $\vec{z} = \vec{r} + b \cdot \vec{w}$ and if $\vec{z} \notin I^n$ for the interval $I = [-(S \cdot B - B); S \cdot B - B]$ he sets $\vec{z} = \perp$ (i.e. he aborts the protocol). Finally he sends $\vec{z}$ to V, who only verifies the response if $\vec{z} \neq \perp$ by checking that $f(\vec{z}) = a \circ y^b$, i.e. $(a, b, \vec{z})$ is an accepting conversation for x.

Likewise, the simulator Sim on input b_s and x would have been defined as follows:

1. Chooses $\vec{z_\mathsf{s}} \in_R \mathbb{Z}^n$ uniformly at random with probability $1 - \Pr[\vec{z} \notin I^n]$ (see Equation (3.1)) such that $\|\vec{z_\mathsf{s}}\|_\infty \leq S \cdot B - B$ (i.e. $\vec{z_\mathsf{s}} \in I^n$), otherwise sets $\vec{z_\mathsf{s}} = \perp$.

2. Computes $a_\mathsf{s} = f(\vec{z_\mathsf{s}}) \circ y^{-b_\mathsf{s}}$ if $\vec{z_\mathsf{s}} \in I^n$, otherwise computes $a_\mathsf{s} = f(\vec{r_\mathsf{s}})$ for some $\vec{r_\mathsf{s}} \in_R \mathbb{Z}^n$, which is chosen uniformly at random such that $\|\vec{r_\mathsf{s}}\|_\infty \leq S \cdot B$.

3. Outputs the conversation $(a_\mathsf{s}, b_\mathsf{s}, \vec{z_\mathsf{s}})$ for x.

The reason why $(a_\mathsf{s}, b_\mathsf{s}, \vec{z_\mathsf{s}})$ may not have the same probability distribution as a conversation $(a_\mathsf{p}, b_\mathsf{p}, \vec{z_\mathsf{p}})$ from the real protocol is because Sim doesn't know the value of P's witness $\vec{w}$. Therefore, if Sim chooses to abort in step (1) of the simulation we can't be sure that $\vec{r_\mathsf{s}} + b_\mathsf{s} \cdot \vec{w} \notin I^n$, which is the only reason why Sim should choose to abort the protocol and where $\vec{r_\mathsf{s}}$ is the one chosen in step (2) of the simulation. This problem is solved by using a commitment scheme to hide the value of a_s and a_p, which implies that Sim doesn't have to choose the randomness $\vec{r_\mathsf{s}}$ in the aborting case because it doesn't have to compute a_s.

The proof for perfect special honest-verifier zero-knowledge. Recall that a conversation from the real protocol is on the form $(com_\mathsf{p}, b_\mathsf{p}, \mathcal{Z}_\mathsf{p})$, and hence the simulator Sim on input x and random challenge bit b_s has to output a conversation $(com_\mathsf{s}, b_\mathsf{s}, \mathcal{Z}_\mathsf{s})$ with the same probability distribution. The simulator Sim is constructed as follows:

1. Chooses to abort the protocol with probability $\Pr[\vec{z} \notin I^n]$ (see Equation (3.1)).

2. If abort:

   (a) Chooses $\vec{d_\mathsf{s}} \in_R \mathbb{Z}^n$ uniformly at random such that $\left\|\vec{d_\mathsf{s}}\right\|_\infty \leq S \cdot B$ and computes $com_\mathsf{s} = \mathsf{commit}_{\mathsf{pk}}(f(d_\mathsf{s}), s_\mathsf{s})$ where $s_\mathsf{s} \in_R \mathbb{Z}$ is the randomness used in the commitment scheme.

   (b) Outputs the conversation $(com_\mathsf{s}, b_\mathsf{s}, \mathcal{Z}_\mathsf{s} = \perp)$.

3. Else no abort:

   (a) Chooses $\vec{z_\mathsf{s}} \in_R \mathbb{Z}^n$ uniformly at random such that $\|\vec{z_\mathsf{s}}\|_\infty \leq S \cdot B - B$, computes $a_\mathsf{s} = f(\vec{z_\mathsf{s}}) \circ y^{-b_\mathsf{s}}$ and $com_\mathsf{s} = \mathsf{commit}_{\mathsf{pk}}(a_\mathsf{s}, s_\mathsf{s})$ where $s_\mathsf{s} \in_R \mathbb{Z}$ is the randomness used in the commitment scheme.

   (b) Outputs the conversation $(com_\mathsf{s}, b_\mathsf{s}, \mathcal{Z}_\mathsf{s} = (\vec{z_\mathsf{s}}, a_\mathsf{s}, s_\mathsf{s}))$.

As shown in the following equation is the protocol still complete when the simulator chooses not to abort:

$$\begin{aligned} f(\vec{z_{\mathsf{s}}}) &= a_{\mathsf{s}} \circ y^{b_{\mathsf{s}}} \\ &= (f(\vec{z_{\mathsf{s}}}) \circ y^{-b_{\mathsf{s}}}) \circ y^{b_{\mathsf{s}}} \\ &= f(\vec{z_{\mathsf{s}}}) \circ y^{-b_{\mathsf{s}}+b_{\mathsf{s}}} \\ &= f(\vec{z_{\mathsf{s}}}) \end{aligned}$$

To prove that the output of the simulator is perfectly indistinguishable from a conversation in the real protocol, i.e. $\mathsf{Sim}(x, b_{\mathsf{s}}) \sim^p (\mathsf{P}(\vec{w}), \mathsf{V})(x)$, we have to prove that the statistical distance between a conversation $(com_{\mathsf{s}}, b_{\mathsf{s}}, \mathcal{Z}_{\mathsf{s}})$ from the simulator and a conversation $(com_{\mathsf{p}}, b_{\mathsf{p}}, \mathcal{Z}_{\mathsf{p}})$ from the real protocol is equal 0. Before we apply DEFINITION 2.7 (the definition of statistical distance) on the simulator and the real protocol we first notice that the challenge bit b_{s} is chosen uniformly at random and then given to the simulator, and hence it has the same probability distribution as the challenge bit b_{p} in the real protocol, i.e. $b_{\mathsf{s}} \sim^p b_{\mathsf{p}}$. Second, since the commitment scheme commit is perfect hiding we have that $com_{\mathsf{s}} \sim^p com_{\mathsf{p}}$. Therefore, in the first case when the simulator chooses to abort, which means that $\mathcal{Z}_{\mathsf{s}} = \perp$ and $\mathcal{Z}_{\mathsf{p}} = \perp$, is $(com_{\mathsf{s}}, b_{\mathsf{s}}, \mathcal{Z}_{\mathsf{s}}) \sim^p (com_{\mathsf{p}}, b_{\mathsf{p}}, \mathcal{Z}_{\mathsf{p}})$ and hence $\mathsf{Sim}(x, b_{\mathsf{s}}) \sim^p (\mathsf{P}(\vec{w}), \mathsf{V})(x)$.

Now, for the second case when then simulator chooses not to abort it has to send the response and open the commitment, and hence we have to argue that $\vec{z_{\mathsf{s}}} \sim^p \vec{z_{\mathsf{p}}}$, $s_{\mathsf{s}} \sim^p s_{\mathsf{p}}$ and $a_{\mathsf{s}} \sim^p a_{\mathsf{p}}$. Since s_{s} and s_{p} are both chosen uniformly at random they have the same probability distribution. The simulator computes a_{s} as $a_{\mathsf{s}} = f(\vec{z_{\mathsf{s}}}) \circ y^{-b_{\mathsf{s}}}$, which means that a_{s} is determined by $\vec{z_{\mathsf{s}}}$ and b_{s}. But as argued before has b_{s} the same probability distribution as b_{p}, and hence all we need to prove is that the statistical distance between $\vec{z_{\mathsf{s}}}$ and $\vec{z_{\mathsf{p}}}$ is 0. Using DEFINITION 2.7 we let P correspond to the simulator and Q to the real protocol. The statistical distance between P and Q is:

$$\begin{aligned} SD(P, Q) &= \sum_{\zeta \neq \perp} |P(\zeta) - Q(\zeta)| \\ &= \sum_{\zeta \neq \perp} |\Pr[\vec{z_{\mathsf{s}}} = \zeta] - \Pr[\vec{z_{\mathsf{p}}} = \zeta]| \end{aligned} \tag{3.2}$$

We then have to argue that Equation (3.2) is equal 0. For $\vec{z_{\mathsf{p}}} \neq \perp$ in the real protocol we know that $\|\vec{z_{\mathsf{p}}}\|_\infty \leq S \cdot B - B$, which is exactly the same interval of size $T = 2 \cdot (S \cdot B - B) + 1$ as the simulator chooses $\vec{z_{\mathsf{s}}}$ from, and hence:

$$\Pr[\vec{z_{\mathsf{s}}} = \zeta] = \Pr[\vec{z_{\mathsf{p}}} = \zeta] = \frac{1}{T} \tag{3.3}$$

Using Equation (3.3) in (3.2) we get the desired result:

$$\begin{aligned} SD(P,Q) &= \sum_{\zeta \neq \perp} |\Pr[\vec{z_{\mathsf{s}}} = \zeta] - \Pr[\vec{z_{\mathsf{p}}} = \zeta]| \\ &= \sum_{\zeta \neq \perp} \left| \frac{1}{T} - \frac{1}{T} \right| \\ &= 0 \end{aligned}$$

We will now argue why the protocol is computational sHVZK when the commitment scheme is unconditional binding and computational hiding. In this case is the only difference in the above proof that $com_{\mathsf{s}} \sim^c com_{\mathsf{p}}$ instead of $com_{\mathsf{s}} \sim^p com_{\mathsf{p}}$ (as defined in Section 2.7), and hence $\mathsf{Sim}(x, b_{\mathsf{s}}) \sim^c (\mathsf{P}(\vec{w}), \mathsf{V})(x)$.

3.2 Comparison of the Aborting and Non-Aborting Version of the General Framework

In this section we first present a non-aborting version of the general framework as presented in Section 3.1 and then we compares the two. The non-aborting version is almost identical to the aborting version, but with the main differences that the interval which the randomness $\vec{r}$ is chosen from is much larger compared to the interval that $\vec{w}$ is chosen from, and we don't have to use a commitment scheme. The function f, witness $\vec{w}$ and problem $x = (f, y)$ are defined as in Section 3.1. The non-aborting version of the general framework is defined as follows:

PROTOCOL 3.2: The non-aborting version of the general framework

1. The prover P chooses $\vec{r}$ uniformly at random from $\mathbb{Z}^n$ such that $\|\vec{r}\|_\infty \leq 2^k \cdot B$ for the parameter $k \geq 1$ and sends $a = f(\vec{r})$ to the verifier V.
2. V chooses a challenge bit b uniformly at random and sends b to P.
3. P computes $\vec{z} = \vec{r} + b \cdot \vec{w}$ and sends $\vec{z}$ to V. Finally V verifies the response by checking that $f(\vec{z}) = a \circ y^b$, i.e. $(a, b, \vec{z})$ is an accepting conversation for x.

Illustration of the protocol

PROTOCOL 3.2: The non-aborting version of the general framework (continued)

Prover $P(\vec{w}, x)$		**Verifier** $V(x)$
$\vec{r} \in_R \mathbb{Z}^n$		
	$\xrightarrow{a = f(\vec{r})}$	
	$\xleftarrow{b \in_R \{0,1\}}$	
	$\xrightarrow{\vec{z} = \vec{r} + b \cdot \vec{w}}$	
		accept iff $f(\vec{z}) = a \circ y^b$

The security of the protocol lies in the fact that adding a small vector $b \cdot \vec{w}$ to a large vector $\vec{r}$ doesn't have much influence on the result. I.e. we can think of the response $\vec{z}$ as $\vec{z} = \vec{r}$ where $\vec{r}$ is just a vector chosen uniformly at random. Therefore, we can think of k as the security parameter of the protocol since it defines how large an interval $\vec{r}$ is chosen from compared to $\vec{w}$. This is used in the proof of THEOREM 3.2 where we have to prove that some statistical distance is negligible small.

THEOREM 3.2. *The non-aborting version of the general framework presented in this section satisfies perfect completeness, perfect special soundness where the extracted witness has size* $\|\vec{w}^*\|_\infty \leq 2 \cdot (2^k \cdot B + B)$ *and statistical special honest-verifier zero-knowledge.*

Proof of Theorem 3.2. The proof for *perfect* completeness and *perfect* special soundness as defined in DEFINITION 2.21 are almost identical to the ones given in the proof of THEOREM 3.1. The only differences are that we don't have to consider the case when the prover P chooses to abort (and hence the use of the commitment scheme, which weakened the special soundness property in the aborting version) and the extracted witness $\vec{w}^*$ from the perfect special soundness proof has size $\|\vec{w}^*\|_\infty \leq 2 \cdot (2^k \cdot B + B)$ because $\|\vec{z}\|_\infty \leq 2^k \cdot B + B$.

Statistical special honest-verifier zero-knowledge (sHVZK) is almost similar to perfect sHVZK as defined in DEFINITION 2.21 except that we have to prove that the output from the simulator Sim is statistically indistinguishable from a conversation in the real protocol. The simulator Sim on input x and random challenge b_s is constructed as follows:

1. Chooses $\vec{z_\mathsf{s}}$ uniformly at random in $\mathbb{Z}^n$ such that $\|\vec{z_\mathsf{s}}\|_\infty \leq 2^k \cdot B + B$.
2. Computes $a_\mathsf{s} = f(\vec{z_\mathsf{s}}) \circ y^{-b_\mathsf{s}}$.
3. Outputs the conversation $(a_\mathsf{s}, b_\mathsf{s}, \vec{z_\mathsf{s}})$ for x.

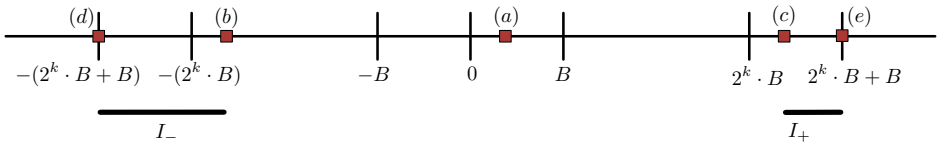

FIGURE 3.2

To prove that $\mathsf{Sim}(x, b_\mathsf{s}) \sim^s (\mathsf{P}(\vec{w}), \mathsf{V})(x)$ we have to argue that the statistical distance between a conversation $(a_\mathsf{s}, b_\mathsf{s}, \vec{z_\mathsf{s}})$ from the simulator and a conversation $(a_\mathsf{p}, b_\mathsf{p}, \vec{z_\mathsf{p}})$ from the real protocol is negligible in k. We use k since it's the security parameter of the protocol as argued previously. Before we use DEFINITION 2.7 (the definition of the statistical distance) we notice that both challenge bits b_s and b_p are distributed equally because both are chosen uniformly at random. Also, the simulator computes a_s as $a_\mathsf{s} = f(\vec{z_\mathsf{s}}) \circ y^{-b_\mathsf{s}}$, which means that a_s is determined by $\vec{z_\mathsf{s}}$ and b_s. Therefore, all we need to prove is that the statistical distance between $\vec{z_\mathsf{s}}$ and $\vec{z_\mathsf{p}}$ is negligible in k. Using DEFINITION 2.7 we let P correspond to the simulator and Q to the real protocol. The statistical distance between P and Q is:

$$\begin{aligned} SD(P,Q) &= \sum_\zeta |P(\zeta) - Q(\zeta)| \\ &= \sum_\zeta |\Pr[\vec{z_\mathsf{s}} = \zeta] - \Pr[\vec{z_\mathsf{p}} = \zeta]| \end{aligned} \tag{3.4}$$

Let $\vec{w}$ be any vector such that $\|\vec{w}\|_\infty \leq B$, e.g. the one at point (a) in FIGURE 3.2. This implies that $\vec{z_\mathsf{p}}$ lies between point (b) and (c) in the figure. However, this is not the same interval as the simulator chooses $\vec{z_\mathsf{s}}$ from, which is between point (d) and (e) in the figure and of size $T = 2 \cdot (2^k \cdot B + B) + 1$. Therefore, in the interval between point (d) and (b), also illustrated by I_- in the figure, is the statistical distance $\sum_{\zeta \in I_-} \left|\frac{1}{T} - 0\right|$. Similar for the interval I_+ in the figure is the statistical distance $\sum_{\zeta \in I_+} \left|\frac{1}{T} - 0\right|$. Because of symmetry is the size of the union of I_- and I_+ equal $|I_-| + |I_+| = 2 \cdot B$. Putting it all together, we get the following result:

$$\begin{aligned} SD(P,Q) &= \sum_\zeta |\Pr[\vec{z_\mathsf{s}} = \zeta] - \Pr[\vec{z_\mathsf{p}} = \zeta]| \\ &= \sum_{\zeta \in I_-} \left|\frac{1}{T} - 0\right| + \sum_{\zeta \notin I_- \cup I_+} |0 - 0| + \sum_{\zeta \in I_+} \left|\frac{1}{T} - 0\right| \\ &= |I_-| \cdot \frac{1}{T} + 0 + \cdots + 0 + |I_+| \cdot \frac{1}{T} \\ &= (|I_-| + |I_+|) \cdot \frac{1}{T} \\ &= (2 \cdot B) \cdot \frac{1}{T} \end{aligned} \tag{3.5}$$

where Equation (3.5) is negligible in k because $T = 2 \cdot (2^k \cdot B + B) + 1$ is exponential large in k.

□

TABLE 3.2: Comparison of the aborting and non-aborting version of the general framework. See PROTOCOL 3.1 for an illustration of the aborting version and PROTOCOL 3.2 for the non-aborting version.

	The aborting version	**The non-aborting version**
Parameters	$\|\vec{w}\|_\infty \leq B$ $\|\vec{r}\|_\infty \leq S \cdot B$ $\|\vec{z}\|_\infty \leq S \cdot B - B$ or $\vec{z} = \perp$	$\|\vec{w}\|_\infty \leq B$ $\|\vec{r}\|_\infty \leq 2^k \cdot B$ $\|\vec{z}\|_\infty \leq 2^k \cdot B + B$
Completeness	Aborts with prob. $\Pr[\vec{z} \notin I^n]$ (Eq. (3.1))	Perfect
Special soundness	Statistical/Perfect and $\|\vec{w}^*\|_\infty \leq 2 \cdot (S \cdot B - B)$	Perfect and $\|\vec{w}^*\|_\infty \leq 2 \cdot (2^k \cdot B + B)$
sHVZK	Perfect/Computational	Statistical

We can now finally compare the two versions of the general framework as we have done in TABLE 3.2. The differences between the two are that the non-aborting version chooses the randomness $\vec{r}$ from a much larger interval and has no aborting option, which implies that it's perfectly complete. However, this come at the expense of the size of the extracted witness $\vec{w}^*$, which is much larger compared to the extracted witness in the aborting version, because the response $\vec{z}$ in the non-aborting version is from a much larger interval. The non-aborting version also satisfies perfect special soundness and statistical sHVZK while the aborting version either satisfies perfect special soundness and computational sHVZK when the used commitment scheme is unconditional binding and computational hiding, or statistical special soundness and perfect sHVZK when the commitment scheme is computational binding and perfect hiding.

The above observation about the size of the extracted witness in the two versions is important if we base the security of the protocol on a lattice problem like the shortest vector problem: Since the knowledge extractor Ext in the non-aborting version is able of computing a *large* witness $\vec{w}^*$ such that $(\vec{w}^*, x) \in R$ after having seen two accepting conversations with different challenges from the prover P, it tells us that there exists a *large* preimage of y. Remember that the preimage of y is P's witness. And since the shortest vector problem is reduced to finding a *small* preimage (and not a large one) as argued in Section 2.3, a malicious verifier V* has a high probability of finding this large preimage of y, and hence break the security of the protocol. However, in the aborting version is this not a problem since Ext computes a *small* witness. And hence, in this version has V* a hard time finding a preimage of y.

Chapter 4

Applications of the General Framework

4.1 Protocols based on the General Framework

In this section we present two applications of the general framework as presented in Chapter 3: Girault's protocol [1, 3] and Lyubashevsky's lattice-based protocol [7]. Both protocols were original presented as identification schemes, which means that they don't have to satisfy the special honest verifier zero-knowledge (sHVZK) property, and hence they don't have to use a commitment scheme. A protocol is an identification scheme if it satisfies completeness, special soundness and witness hiding. The witness hiding property says that a possible malicious verifier V^* should not learn anything about the provers witness after executing the protocol (while the sHVZK property says that a honest verifier should not learn any information whatsoever except that the provers claim is true). Hence, the special soundness and witness hiding properties implies that V^*, who first play the role of the verifier with the prover P, can't impersonate P later on with another verifier.

In Girault's protocol is the function $f : \mathbb{Z} \to (\mathbb{Z}_n^*, \cdot)$ defined as $f(c) = g^c \bmod n$ for some $c \in \mathbb{Z}$ where $g \in \mathbb{Z}_n^*$ such that its order $\mathrm{ord}(g)$ is maximal in $\mathbb{Z}_n^*$ and $n = p \cdot q$ for the primes p and q. The witness w is an integer from the domain $\mathbb{Z}$ and the problem x is defined as $x = (f, y)$ where $y = f(w)$. In this protocol is the two primes p and q both unknown for the prover and the verifier, and hence we have to use the general framework because the prover is not able of choosing the randomness from the order of the group $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid \gcd(x, \phi(n)) = 1\}$ where $\phi(n) = (p-1) \cdot (q-1)$.

We can either use the aborting or non-aborting version of the general framework for Girault's protocol. To prove that it's an application of the general framework we only need to define a commitment scheme commit and prove that f is an additive homomorphic function: Let commit be any unconditional binding and computational hiding or computational binding and perfect hiding commitment scheme with public key pk. Furthermore, let $c, d \in \mathbb{Z}$. We have that:

$$\begin{aligned} f(c+d) &= g^{c+d} \bmod n \\ &= (g^c \cdot g^d) \bmod n \\ &= (g^c \bmod n) \cdot (g^d \bmod n) \\ &= f(c) \cdot f(d) \end{aligned}$$

which is the definition of an additive homomorphic function.

In Lyubashevsky's lattice-based protocol are ideal lattices represented as ideals in the ring $L = \mathbb{Z}_p[x]/\langle x^n + 1\rangle$ where p is some odd positive integer and n is any power of 2. Therefore, when we talk about multiplying two vectors we actually first convert them into polynomials in L and then multiply them together. The function $f : D^m \to (L, +, \cdot)$ in Lyubashevsky's protocol is chosen uniformly at random from the family of hash functions $\mathsf{H}(L, D, m)$ (see Section 2.3), i.e. $f(\hat{d}) = \hat{c} \cdot \hat{d} = \vec{c_1} \cdot \vec{d_1} + \cdots + \vec{c_m} \cdot \vec{d_m}$ where $\hat{c} \in L^m$ and $\hat{d} \in D^m$. We can think of the function f as a mapping from the set D^m of lattices of length m to the set L of vectors of length n where $D \subseteq L$. The witness $\hat{w}$ is a lattice from the domain D^m and the problem x is defined as $x = (f, y)$ where $y = f(\hat{w})$. Since the security of the protocol is based on the shortest vector problem we have to use the aborting version of the general framework as argued in Section 3.2.

Again, to prove that Lyubashevsky's protocol is an application of the general framework with abort we only need to define a commitment scheme commit and prove that f is an additive homomorphic function: Let commit be any unconditional binding and computational hiding or computational binding and perfect hiding commitment scheme with public key pk. Furthermore, let $\hat{d}, \hat{e} \in D^m$. We have that:

$$\begin{aligned} f(\hat{d} + \hat{e}) &= \hat{c} \cdot (\hat{d} + \hat{e}) \\ &= (\hat{c} \cdot \hat{d}) + (\hat{c} \cdot \hat{e}) \\ &= f(\hat{d}) + f(\hat{e}) \end{aligned}$$

which is the definition of an additive homomorphic function.

4.2 A Statistically Secure Sigma Protocol in the Standard Model based on the General Framework with Abort

The main reason why the general framework with abort presented in Section 3.1 is not a statistically secure sigma protocol, is because we have not defined what should happen when the prover P chooses to abort, the challenge is only a single bit and a honest P who know a correct witness can be rejected by the verifier V with too high probability. The last argument comes from the fact that we want S to be as small as possible, but this implies that the probability that P chooses to abort with (see Equation (3.1)) gets higher since it depends on S. Therefore, to construct a statistically secure sigma protocol, as illustrated in PROTOCOL 4.1, we execute the general framework with abort t times in parallel where we use a linear secret sharing code $\mathsf{C} = [t, k, d]_2$ with minimum distance $d > 2 \cdot (t - E)$ (see Equation (4.1) for the definition of E) to compute a t-bit codeword, which P has to answer. The verifier V then outputs accept if and only if at least E of the t conversations are accepted. The reason why we use the code C is for the protocol to satisfy statistical special soundness as we will argue in the proof of THEOREM 4.2. As described in Section 2.4 exists there a linear secret sharing code $\mathsf{C} = [t, k, d]_2$ with $(d^{\perp} - 2)$-privacy (when using $\ell = 1$)

and $(t - d + 1)$-reconstruction where $d^{\perp}$ is the minimum distance of the dual code $\mathsf{C}^{\perp} = [t, t-k, d^{\perp}]_2$.

In the protocol is the function f, witness $\vec{w}$, problem $x = (f, y)$ where $y = f(\vec{w})$ and commitment scheme commit with public key pk defined as in the general framework with abort in Section 3.1. The statistically secure sigma protocol in the standard model is defined as follows:

PROTOCOL 4.1: A statistically secure sigma protocol

1. P chooses $\vec{r_1}, \ldots, \vec{r_t}$ uniformly at random from $\mathbb{Z}^n$ such that $\|\vec{r_i}\|_\infty \leq S \cdot B$ for $S \geq 1$. He then computes $a_i = f(\vec{r_i})$ and $com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$ for $i = 1, \ldots, t$ where $s_i \in_R \mathbb{Z}$ is the randomness used in the commitment scheme and sends $(com_1, \ldots, com_t)$ to V.

2. V chooses a k-bit string e uniformly at random and sends e to P.

3. P first use the secret sharing code $\mathsf{C} = [t, k, d]_2$ with minimum distance $d > 2 \cdot (t - E)$ to compute the t-bit codeword $c = \mathsf{C}(e)$. He then computes $\vec{z_i} = \vec{r_i} + c \cdot \vec{w}$ for $i = 1, \ldots, t$. If $\vec{z_i} \in I^n$ for the interval $I = [-(S \cdot B - B); S \cdot B - B]$ he sets $\mathcal{Z}_i = (\vec{z_i}, a_i, s_i)$, otherwise he sets $\mathcal{Z}_i = \perp$. Finally P sends $(\mathcal{Z}_1, \ldots, \mathcal{Z}_t)$ to V who first use C to compute $c = \mathsf{C}(e)$ and then outputs accept if and only if at least E of the t conversations are accepted, i.e. at least E of the t conversations satisfies $\mathcal{Z}_i \neq \perp$, $com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$ and $f(\vec{z_i}) = a_i \circ y^c$.

Illustration of the protocol

PROTOCOL 4.1: A statistically secure sigma protocol (continued)

Prover $P(\vec{w}, x)$		**Verifier** $V(x)$
$\vec{r_i} \in_R \mathbb{Z}^n$		
$a_i = f(\vec{r_i})$		
$s_i \in_R \mathbb{Z}$		
$com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$		
	$\xrightarrow{(com_1, \ldots, com_t)}$	
	$\xleftarrow{e \in_R \{0,1\}^k}$	
$c = \mathsf{C}(e)$		
$\vec{z_i} = \vec{r_i} + c \cdot \vec{w}$		
if $\vec{z_i} \in I^n$ then		
$\mathcal{Z}_i = (\vec{z_i}, a_i, s_i)$		
else $\mathcal{Z}_i = \perp$		
	$\xrightarrow{(\mathcal{Z}_1, \ldots, \mathcal{Z}_t)}$	
		$c = \mathsf{C}(e)$
		accept iff at least E :
		$\mathcal{Z}_i \neq \perp$,
		$com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$
		and $f(\vec{z_i}) = a_i \circ y^c$

The limit E is defined such that a malicious prover P^* who doesn't know a correct witness to the problem x has a hard time convincing V to output accept, while a honest prover P has not. We have that a honest prover will approximately send $t \cdot \Pr[\vec{z} \notin I^n]$ abort messages to the verifier (see Equation (3.1) for the definition of $\Pr[\vec{z} \notin I^n]$), and hence we set E equal the number of expected accepted conversations minus some allowed error probability $\epsilon \in (0; 1]$:

$$E = t \cdot (1 - \Pr[\vec{z} \notin I^n]) - t \cdot \epsilon \tag{4.1}$$

Using the Chernoff-Hoeffding bound as described in Section 2.2 can we prove that the actually number of accepted conversations are at most E with probability negligible in t (i.e. the actually number of accepted conversations are at least E with high probability). A conversation consists of the three messages $(com_i, c, \mathcal{Z}_i)$ (actually, then a conversation consists of the three messages $(com_i, e, \mathcal{Z}_i)$, but since the prover has to answer the codeword c instead of e we use c as the second message) where com_i and $\mathcal{Z}_i$ for $i = 1, \ldots, t$ are fully independent because of the used randomness while c is only $(d^{\perp} - 2)$-wise independent (because we use a code with $(d^{\perp} - 2)$-privacy to generate c). Therefore, we can't use the Chernoff-Hoeffding bound because it requires that all three messages are fully independent, but fortunately can we use the Chernoff-Hoeffding bound with limited independence as described in Section 2.2.1.

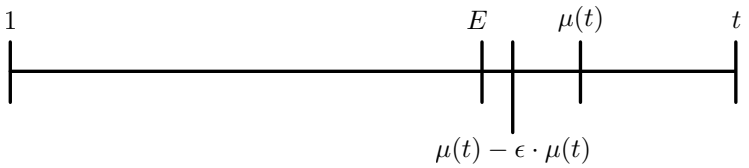

FIGURE 4.1

Let X_i for $i = 1, \ldots, t$ denote the conversations where $X_i = 1$ if conversation i is an accepting conversation and $X_i = 0$ otherwise. Now, $X = \sum_{i=1}^{t} X_i$ denotes the actually number of accepted conversations and

$$\begin{aligned}\mu(t) &= \sum_{i=1}^{t} \mathbb{E}[X_i = 1] \\ &= \sum_{i=1}^{t} (1 - \Pr[\vec{z} \notin I^n]) \\ &= t \cdot (1 - \Pr[\vec{z} \notin I^n]) \end{aligned} \tag{4.2}$$

denotes the expected number of accepted conversations. Using the Chernoff-Hoeffding bound with limited independence we want to prove that X lies between 1 and E in FIGURE 4.1 with probability negligible in t.

Let $d^{\perp} = t \cdot \alpha$ for some $\alpha \in [0; 1]$ and define the independence as $\ell(t) = (t \cdot \alpha) - 2$. Now, using the same $\epsilon \in (0; 1]$ as in Equation (4.1) the Chernoff-Hoeffding bound with limited independence says that if $\ell(t) \leq \left\lfloor \epsilon^2 \cdot \mu(t) \cdot \exp\left(-\frac{1}{3}\right) \right\rfloor$ then

$$\Pr[|X - \mu(t)| \geq \epsilon \cdot \mu(t)] \leq \exp\left(-\left\lfloor \frac{\ell(t)}{2} \right\rfloor\right)$$

and if $\ell(t) = \left\lfloor \epsilon^2 \cdot \mu(t) \cdot \exp\left(-\frac{1}{3}\right) \right\rfloor$ then

$$\Pr[|X - \mu(t)| \geq \epsilon \cdot \mu(t)] \leq \exp\left(-\left\lfloor \frac{\epsilon^2 \cdot \mu(t)}{3} \right\rfloor\right)$$

where $\exp(x) = e^x$. In other words, the probability that the actual number of accepted conversations X deviates with more than $\epsilon \cdot \mu(t)$ from the expected number of accepted conversation $\mu(t)$ is negligible in t. And hence, the probability that X lies between 1 and $\mu(t) - \epsilon \cdot \mu(t)$ in FIGURE 4.1 is negligible in t. Therefore, if we can argue that the distance between E and $\mu(t)$ is larger than or equal to the distance between $\mu(t) - \epsilon \cdot \mu(t)$ and $\mu(t)$, we have proved that X lies between 1 and E with probability negligible in t. And because the verifier only outputs accept if he has received at least E accepting conversations, the verifier outputs reject with probability negligible in t for a honest prover. To prove that $|E - \mu(t)| \geq \epsilon \cdot \mu(t)$ we use the

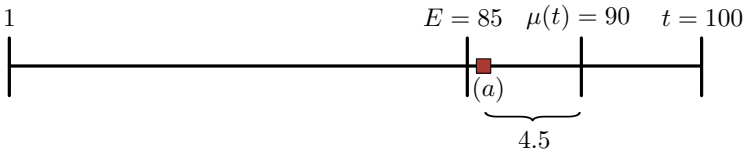

FIGURE 4.2

fact that $t \geq \mu(t)$, Equation (4.1) and (4.2):

$$\begin{aligned} |E - \mu(t)| &= |(t \cdot (1 - \Pr[\vec{z} \notin I^n]) - t \cdot \epsilon) - \mu(t)| \\ &= |(\mu(t) - t \cdot \epsilon) - \mu(t)| \\ &= |-t \cdot \epsilon| \\ &= t \cdot \epsilon \\ &\geq \mu(t) \cdot \epsilon \end{aligned}$$

EXAMPLE 4.1. Assume that $t = 100$, $\Pr[\vec{z} \notin I^n] = 0.1$ and $\epsilon = 0.05$. If the verifier V executes PROTOCOL 4.1 with a honest prover P, he would expect to receive $t \cdot \Pr[\vec{z} \notin I^n] = 10$ abort messages, and hence after executing the protocol V outputs accept if and only if at least

$$E = t \cdot (1 - \Pr[\vec{z} \notin I^n]) - t \cdot \epsilon = 85$$

of the conversations are accepted. Using $\mu(t) = t \cdot (1 - \Pr[\vec{z} \notin I^n]) = 90$ in the Chernoff-Hoeffding bound with limited independence we get the following probability, which is negligible in t:

$$\Pr[|X - \mu(t)| \geq \epsilon \cdot \mu(t)] = \Pr[|X - 90| \geq 4.5]$$

In other words, the probability that X deviates with more than 4.5 from $\mu(t) = 90$ is negligible in t, which implies that X lies between 1 and point (a) in FIGURE 4.2 with probability negligible in t. And hence, X lies between 1 and $E = 85$ with probability negligible in t.

$\Diamond$

THEOREM 4.2. *Let* $\mathsf{commit}^{ub,ch}$ *be an unconditional binding and computational hiding commitment scheme and* $\mathsf{commit}^{cb,ph}$ *a computational binding and perfect hiding commitment scheme.*

The protocol presented in this section satisfies statistical completeness, special soundness (perfect if $\mathsf{commit}^{ub,ch}$ *is used and statistical if* $\mathsf{commit}^{cb,ph}$ *is used) and special honest-verifier zero-knowledge (computational if* $\mathsf{commit}^{ub,ch}$ *is used and perfect if* $\mathsf{commit}^{cb,ph}$ *is used) in the standard model, and hence is a statistically secure sigma protocol.*

Proof of Theorem 4.2. Let (P, V) be the general framework with abort presented in Section 3.1 for the relation $(\vec{w}, x) \in R$. The prover P and the

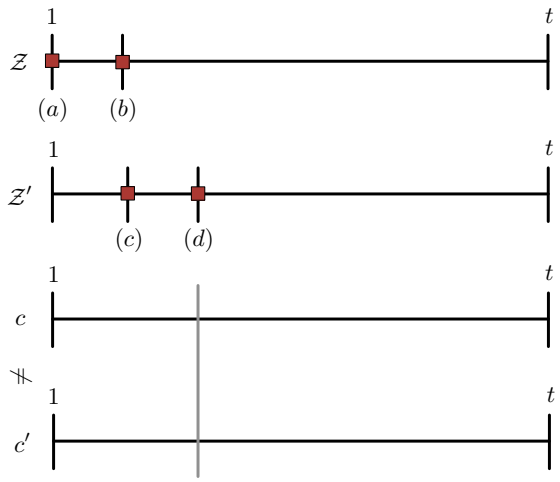

FIGURE 4.3

verifier V are both given x as common input and P is given $\vec{w}$ as private input. Furthermore, let $(\mathsf{P}_\Sigma, \mathsf{V}_\Sigma)$ be the protocol presented in this section for the same relation R where P_Σ is the prover and V_Σ is the verifier. If $(\mathsf{P}_\Sigma, \mathsf{V}_\Sigma)$ is a statistically secure sigma protocol it has to satisfy statistical completeness, statistical special soundness and computational special honest-verifier zero-knowledge as defined in DEFINITION 2.23.

Using the above argument about how E is defined and that the actual number of accepting conversations lies between 1 and E with probability negligible in t, we have that $(\mathsf{P}_\Sigma, \mathsf{V}_\Sigma)$ satisfies statistical completeness. Furthermore, since (P, V) satisfies computational special honest-verifier zero-knowledge (sHVZK) we have that $(\mathsf{P}_\Sigma, \mathsf{V}_\Sigma)$ also satisfies this property because sHVZK is invariant under parallel composition.

To prove that $(\mathsf{P}_\Sigma, \mathsf{V}_\Sigma)$ satisfies statistical special soundness we argue that if a possible malicious prover P^*_Σ can produce two accepting conversations $(com, c, \mathcal{Z})$ and $(com', c', \mathcal{Z}')$ with different challenges $c \neq c'$, there exists an index j such that $c_j \neq c'_j$ for the conversations $(com_j, c_j, \mathcal{Z}_j)$ and $(com'_j, c'_j, \mathcal{Z}'_j)$. And since this is two conversations on the same form as in (P, V), which satisfies statistical special soundness, we have that $(\mathsf{P}_\Sigma, \mathsf{V}_\Sigma)$ also satisfies statistical special soundness.

Assume that P^*_Σ can produce two accepting conversations $(com, c, \mathcal{Z})$ and $(com', c', \mathcal{Z}')$ with different challenges $c \neq c'$. Remember that it's just the general framework with abort executed t times in parallel with at least E accepting conversations where P^*_Σ has correctly answered $c = \mathsf{C}(e)$ and $c' = \mathsf{C}(e')$. Hence, c and c' are both t-bit strings and $\mathcal{Z}$ and $\mathcal{Z}'$ both contains the t answers.

We know that at most $t - E$ of both $\mathcal{Z}$ and $\mathcal{Z}'$ are aborting answers. Assume that these aborting answers are $\mathcal{Z}_i = \perp$ for all i between point (a) and (b) in FIGURE 4.3 and $\mathcal{Z}'_i = \perp$ for all i between point (c) and (d) in the figure. Therefore, if the Hamming distance between two codewords is $\Delta(c, c') > (t - E) + (t - E) = 2 \cdot (t - E)$ (as illustrated with the grey line in the figure), there exists in worst case at least one index j where P^*_Σ has produced two accepting conversations $(com_j, c_j, \mathcal{Z}_j)$ and $(com'_j, c'_j, \mathcal{Z}'_j)$

with different challenges $c_j \neq c'_j$. And since the used code $\mathsf{C} = [t, k, d]_2$ has minimum distance $d > 2 \cdot (t - E)$ we have that $\Delta(c, c') > 2 \cdot (t - E)$ for all possible challenges $c, c' \in \mathsf{C}$. To make sure that $\Delta(c, c') > 2 \cdot (t - E)$ we just chooses E such that $d > 2 \cdot (t - E)$.

□

4.3 A Statistically Secure Non-Interactive Sigma Protocol in the Random Oracle Model based on the General Framework with Abort

To construct a statistically secure non-interactive sigma protocol in the random oracle model we use the Fiat-Shamir transformation as described in Section 2.11 on the statistically secure sigma protocol presented in the previous section. However, as illustrated in the following exists there two versions depending on whether we want an efficient protocol that satisfies statistical completeness as defined in DEFINITION 2.23, or a less efficient one that satisfies perfect completeness as defined in DEFINITION 2.21. In both versions are the function f, witness $\vec{w}$, problem $x = (f, y)$ where $y = f(\vec{w})$, code $\mathsf{C} = [t, k, d]_2$ with minimum distance $d > 2 \cdot (t - E)$ and commitment scheme commit with public key pk defined as in the statistically secure sigma protocol in Section 4.2. The version that satisfies *statistical* completeness is presented in PROTOCOL 4.2 and the one that satisfies *perfect* completeness is presented in PROTOCOL 4.3.

PROTOCOL 4.2: A statistically secure non-interactive sigma protocol in the ROM satisfying *statistical* completeness

1. The prover P chooses $\vec{r_1}, \ldots, \vec{r_t}$ uniformly at random from $\mathbb{Z}^n$ such that $\|\vec{r_i}\|_\infty \leq S \cdot B$ for $S \geq 1$. He then computes $a_i = f(\vec{r_i})$ for $i = 1, \ldots, t$ and sends $com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$ to the random oracle $\mathcal{O}$, which return a k-bit challenge e. $s_i \in_R \mathbb{Z}$ is the randomness used in the commitment scheme. P then use the secret sharing code $\mathsf{C} = [t, k, d]_2$ with minimum distance $d > 2 \cdot (t - E)$ to compute the t-bit codeword $c = \mathsf{C}(e)$. He then computes $\vec{z_i} = \vec{r_i} + c \cdot \vec{w}$ for $i = 1, \ldots, t$. If $\vec{z_i} \in I^n$ for the interval $I = [-(S \cdot B - B); S \cdot B - B]$ he sets $\mathcal{Z}_i = (\vec{z_i}, a_i, s_i)$, otherwise he sets $\mathcal{Z}_i = \perp$. Finally P sends $((com_1, \ldots, com_t), (\mathcal{Z}_1, \ldots, \mathcal{Z}_t))$ to the verifier V.

2. After receiving $((com_1, \ldots, com_t), (\mathcal{Z}_1, \ldots, \mathcal{Z}_t))$ V first sends $(com_1, \ldots, com_t)$ to $\mathcal{O}$, which return the k-bit challenge e, and then use the secret sharing code C to compute $c = \mathsf{C}(e)$. Finally V outputs accept if and only if at least E of the t conversations are accepted, i.e. at least E of the t conversations satisfies $\mathcal{Z}_i \neq \perp$, $com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$ and $f(\vec{z_i}) = a_i \circ y^c$.

Illustration of the protocol

PROTOCOL 4.2: A statistically secure non-interactive sigma protocol in the ROM satisfying *statistical* completeness (continued)

Prover $P(\vec{w}, x)$

$\vec{r_i} \in_R \mathbb{Z}^n$
$a_i = f(\vec{r_i})$
$s_i \in_R \mathbb{Z}$
$com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$
$e = \mathcal{O}(com_1, \ldots, com_t)$
$c = \mathsf{C}(e)$
$\vec{z_i} = \vec{r_i} + c \cdot \vec{w}$
if $\vec{z_i} \in I^n$ then
  $\mathcal{Z}_i = (\vec{z_i}, a_i, s_i)$
else $\mathcal{Z}_i = \perp$

$(com_1, \ldots, com_t),$
$(\mathcal{Z}_1, \ldots, \mathcal{Z}_t)$ $\longrightarrow$

Verifier $V(x)$

$e = \mathcal{O}(com_1, \ldots, com_t)$
$c = \mathsf{C}(e)$
accept iff at least E :
  $\mathcal{Z}_i \neq \perp$,
  $com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$
  and $f(\vec{z_i}) = a_i \circ y^c$

PROTOCOL 4.3: A statistically secure non-interactive sigma protocol in the ROM satisfying *perfect* completeness

1. This step is similar to step (1) in PROTOCOL 4.2 except that if $\vec{z_i} \notin I^n$ then P repeats step (1) with some new randomness $\vec{r_i}' \neq \vec{r_i}$ until $\vec{z_i} \in I^n$.
2. This step is similar to step (2) in PROTOCOL 4.2 except that V only outputs accept if an only if all t conversations are accepted, i.e. all t conversations satisfies $com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$ and $f(\vec{z_i}) = a_i \circ y^c$.

Illustration of the protocol

PROTOCOL 4.3: A statistically secure non-interactive sigma protocol in the ROM satisfying *perfect* completeness (continued)

Prover $P(\vec{w}, x)$		**Verifier** $V(x)$
$\vec{r_i} \in_R \mathbb{Z}^n$		
$a_i = f(\vec{r_i})$		
$s_i \in_R \mathbb{Z}$		
$com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$		
$e = \mathcal{O}(com_1, \ldots, com_t)$		
$c = \mathsf{C}(e)$		
$\vec{z_i} = \vec{r_i} + c \cdot \vec{w}$		
if $\vec{z_i} \in I^n$ then		
$\mathcal{Z}_i = (\vec{z_i}, a_i, s_i)$		
else try again		
	$(com_1, \ldots, com_t),$ $(\mathcal{Z}_1, \ldots, \mathcal{Z}_t)$ $\longrightarrow$	
		$e = \mathcal{O}(com_1, \ldots, com_t)$
		$c = \mathsf{C}(e)$
		accept iff all t :
		$com_i = \mathsf{commit}_{\mathsf{pk}}(a_i, s_i)$
		and $f(\vec{z_i}) = a_i \circ y^c$

Chapter 5

Conclusion

In this work we have presented a general framework for protocols with abort that satisfies completeness with abort, statistical special soundness and computational honest verifier zero-knowledge. We compared the aborting version of the framework with a non-aborting version, and it showed how important the abort technique is when we base the security on lattice problems like the shortest vector problem: In the non-aborting version can a malicious verifier compute a correct witness with high probability because he only has to find a large preimage. However, in the aborting version has the malicious verifier a hard time since he has to find a small preimage. Using the framework as a building block we have constructed a statistically secure sigma protocol that satisfies statistical completeness, statistical special soundness and computational honest verifier zero-knowledge. This new kind of sigma protocol is useful in addition to that we can base the security on lattice problems, when we want the response that the prover sends to the verifier in the third steps to be small, e.g. in a signature scheme.

As stated previously satisfies the statistically secure sigma protocol either statistical special soundness and perfect honest verifier zero-knowledge or perfect special soundness and computational honest verifier zero-knowledge. Therefore, it could be interesting to investigate whether both properties could be perfect at the same time, i.e. have both perfect special soundness and perfect honest verifier zero-knowledge, which is the same two requirements that a sigma protocol has to satisfy. However, this requires that another technique than a commitment scheme is used since it weaken one of the two properties depending on the flavor of the commitment scheme.

List of Protocols

PROTOCOL 3.1: The general framework with abort 22
PROTOCOL 3.2: The non-aborting version of the general framework 29
PROTOCOL 4.1: A statistically secure sigma protocol 35
PROTOCOL 4.2: A statistically secure non-interactive sigma protocol in the ROM satisfying *statistical* completeness . . . 40
PROTOCOL 4.3: A statistically secure non-interactive sigma protocol in the ROM satisfying *perfect* completeness 41

Bibliography

[1] Marc Girault. "Self-certified Public Keys". In: *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT'91. Brighton, UK: Springer-Verlag, 1991, pp. 490–497. ISBN: 3-540-54620-0. URL: http://dl.acm.org/citation.cfm?id=1754868.1754923.

[2] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. "Chernoff-Hoeffding Bounds for Applications with Limited Independence". In: *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '93. Austin, Texas, USA: Society for Industrial and Applied Mathematics, 1993, pp. 331–340. ISBN: 0-89871-313-7. URL: http://dl.acm.org/citation.cfm?id=313559.313797.

[3] David Pointcheval. "The Composite Discrete Logarithm and Secure Authentication". In: *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*. PKC '00. London, UK, UK: Springer-Verlag, 2000, pp. 113–128. ISBN: 3-540-66967-1. URL: http://dl.acm.org/citation.cfm?id=648117.746596.

[4] David Pointcheval and Jacques Stern. "Security Arguments for Digital Signatures and Blind Signatures". In: *J. Cryptol.* 13.3 (Jan. 2000), pp. 361–396. ISSN: 0933-2790. DOI: 10.1007/s001450010003. URL: http://dx.doi.org/10.1007/s001450010003.

[5] Vadim Lyubashevsky and Daniele Micciancio. "Generalized Compact Knapsacks Are Collision Resistant". In: *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*. ICALP'06. Venice, Italy: Springer-Verlag, 2006, pp. 144–155. ISBN: 3-540-35907-9, 978-3-540-35907-4. DOI: 10.1007/11787006_13. URL: http://dx.doi.org/10.1007/11787006_13.

[6] Hao Chen et al. "Advances in Cryptology - EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Proceedings". In: ed. by Moni Naor. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. Chap. Secure Computation from Random Error Correcting Codes, pp. 291–310. ISBN: 978-3-540-72540-4. DOI: 10.1007/978-3-540-72540-4_17. URL: http://dx.doi.org/10.1007/978-3-540-72540-4_17.

[7] Vadim Lyubashevsky. "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures". In: *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. ASIACRYPT '09. Tokyo, Japan: Springer-Verlag, 2009, pp. 598–616. ISBN: 978-3-642-10365-0. DOI: 10.1007/978-3-642-10366-7_35. URL: http://dx.doi.org/10.1007/978-3-642-10366-7_35.

[8] G. Barthe et al. "A Machine-Checked Formalization of Sigma-Protocols". In: *Computer Security Foundations Symposium (CSF), 2010 23rd IEEE*. 2010, pp. 246–260. DOI: 10.1109/CSF.2010.24.

[9] Ivan Damgård. *On Sigma-protocols*. Lecture Notes in Computer Science. Aarhus University, 2011. URL: http://www.cs.au.dk/~ivan/Sigma.pdf.

[10] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. *Solving Hard Lattice Problems and the Security of Lattice-Based Cryptosystems*. Cryptology ePrint Archive, Report 2012/533. http://eprint.iacr.org/2012/533. 2012.

[11] Ivan Damgård and Jesper Buus Nielsen. *Commitment Schemes and Zero-Knoweldge Protocols*. Lecture Notes in Computer Science. Aarhus University, 2014. URL: http://www.daimi.au.dk/~ivan/ComZK08.pdf.

[12] Anders Fog Bunzel. "Sigma Protocols with Abort". Paper from a course at Aarhus University. 2015.