Compressed Σ-protocol theory

Attema, T.

CHAPTER 2

2 Preliminaries

2.1 Basic Notation

We first introduce the basic notation used throughout this dissertation. For a more detailed introduction to concepts such as groups, rings, fields, ideals, modules, homomorphisms, endomorphisms and tensor products, we refer the reader to textbooks such as [Lan02].

By $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ we denote the set of the positive integers, the integers, the real numbers and the nonnegative real numbers, respectively. We write $[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}$ for the set of real numbers bounded by a and b. For a set S, $2^S = \{A \subseteq S\}$ denotes the powerset of S, containing all subsets of S. Moreover, we adhere to the convention in cryptography by defining $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ as the ring of integers modulo $q \in \mathbb{Z}$, i.e., $\mathbb{Z}_q$ does not refer to the ring of q-adic integers. Oftentimes q is prime, in which case the ring $\mathbb{Z}_q$ is a field.

The set of bitstrings of length $n \in \mathbb{N}$ is denoted as $\{0,1\}^n$. Moreover, $|x|$ denotes the length of a bitstring x, i.e., $|x| = n$ for all $x \in \{0,1\}^n$. The set of arbitrarily long bitstrings is denoted as $\{0,1\}^* = \cup_{n\in\mathbb{N}}\{0,1\}^n$. Further, vectors $\mathbf{x} = (x_1, \ldots, x_n)$ are written in boldface.

A group $\mathbb{G}$ with group operation $+$ is denoted as $(\mathbb{G}, +)$. If the group operation is clear from context we simply write $\mathbb{G}$. All groups in this work are assumed to be abelian, i.e., the group operation is commutative. The group of homomorphisms from $\mathbb{G}$ to $\mathbb{H}$ is denoted as $\mathrm{Hom}(\mathbb{G}, \mathbb{H})$. Its group operation is defined as the addition of homomorphisms, i.e., $f + g \colon \mathbb{G} \to \mathbb{H}$, $x \mapsto f(x) + g(x)$ for $f, g \in \mathrm{Hom}(\mathbb{G}, \mathbb{H})$. The set $\mathrm{End}(\mathbb{G}) := \mathrm{Hom}(\mathbb{G}, \mathbb{G})$ contains the endomorphisms of $\mathbb{G}$. The composition of homomorphisms defines a second binary operation (multiplication), i.e., $\mathrm{End}(\mathbb{G})$ is a ring.[1]

Sometimes we use multiplicative notation for the group operation instead and write $(\mathbb{H}, \cdot)$. If the group operation is written additively we denote the identity element by 0, and if the group operation is written multiplicatively we denote the identity element by 1.

Recall that an abelian group $(\mathbb{G}, +)$ is a $\mathbb{Z}$-module, i.e., it has a well-defined

[1] Every ring is defined to contain a multiplicative unit, and all ring homomorphisms are defined to map the multiplicative unit to the multiplicative unit.

multiplication by integers operation

$$\cdot\colon \mathbb{Z} \times \mathbb{G} \to \mathbb{G}, \quad (a, g) \mapsto a \cdot g\,.$$

More generally, let $\mathcal{R}$ be a commutative ring, then an $\mathcal{R}$-module is an abelian group $(\mathbb{G}, +)$ together with a ring homomorphism

$$\phi\colon \mathcal{R} \to \mathrm{End}(\mathbb{G}), \quad a \mapsto \phi_a\,.$$

In particular, the multiplication of $g \in \mathbb{G}$ by $a \in \mathcal{R}$ is defined as $a \cdot g := \phi_a(g)$. Further, $M \otimes_{\mathcal{R}} N$ denotes the tensor product of two $\mathcal{R}$-modules M and N.

The exponent q of an abelian group $(\mathbb{G}, +)$ is the smallest positive integer $q \in \mathbb{N}$, such that $q \cdot g = 0$ for all $g \in \mathbb{G}$. If no such integer q exist, we define $q = \infty$. It is easily seen that an abelian group $(\mathbb{G}, +)$ with exponent q is a $\mathbb{Z}_q$-module.

Let now $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{H}, \cdot)$ be groups of prime order q, hence they are $\mathbb{Z}_q$-modules. Then a mapping $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{H}$ is said to be a *pairing* if it is bilinear, nondegenerate (i.e., e is not identically equal to the identity) and there exists an efficient algorithm to compute e. The tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, e)$ is also referred to as a *bilinear group*.

Finally, we recall the definitions of negligible and noticeable functions.

Definition 2.1 (Negligible Function)**.** A function $\eta\colon \mathbb{N} \to \mathbb{R}$ is said to be negligible, denoted by $\eta(\lambda) \leq \mathrm{negl}(\lambda)$, if for all $c \in \mathbb{N}$ there exists an $N_c \in \mathbb{N}$ such that $|\eta(\lambda)| \leq \lambda^{-c}$ for all $\lambda \geq N_c$.

Definition 2.2 (Noticeable Function)**.** A function $\eta\colon \mathbb{N} \to \mathbb{R}$ is said to be noticeable if there exists a $c \in \mathbb{N}$ and $N \in \mathbb{N}$ such that $|\eta(\lambda)| \geq \lambda^{-c}$ for all $\lambda \geq N$.

These definitions have straightforward adaptations to functions $\eta\colon \{0,1\}^* \to \mathbb{R}$ taking arbitrary bitstrings as input. For instance, a function $\eta\colon \{0,1\}^* \to \mathbb{R}$ is said to be negligible if for all $c \in \mathbb{N}$ there exists an $N_c \in \mathbb{N}$ such that $|\eta(x)| \leq |x|^{-c}$ for all $|x| \geq N_c$.

2.2 Algorithms

Given a probabilistic algorithm $\mathcal{A}$, we write $y = \mathcal{A}(x; r)$ for the output produced by $\mathcal{A}$ on input x and randomness r. Sometimes the randomness is left implicit, i.e., we write $y \leftarrow \mathcal{A}(x)$ for the process of sampling the bits in r uniformly at random and evaluating $y = \mathcal{A}(x; r)$. The randomness r is also referred to as the random coins or the random tape of $\mathcal{A}$. Note that a function is simply a deterministic algorithm. An algorithm is said to be *efficient* or *polynomial time* if $\mathcal{A}(x)$ runs in a number of steps that is polynomial in the input size $|x|$.

Definition 2.3 (Polynomial Time Algorithm)**.** An algorithm $\mathcal{A}$ is a (strict) polynomial time algorithm if there exists a polynomial $p \in \mathbb{Z}[X]$ such that, for all inputs x and random coins r, $\mathcal{A}(x; r)$ runs in at most $p(|x|)$ steps.

The following weaker, but oftentimes sufficient, notion of efficiency only requires $\mathcal{A}(x)$ to run in a polynomial number of steps on *expectation* over the algorithm's randomness.

Definition 2.4 (Expected Polynomial Time Algorithm)**.** An algorithm $\mathcal{A}$ is an expected polynomial time algorithm if there exists a polynomial $p \in \mathbb{Z}[X]$ such that, for all inputs x, $\mathcal{A}(x)$ runs in an expected number of at most $p(|x|)$ steps, where the expectation is over the randomness r of $\mathcal{A}$.

An algorithm $\mathcal{B}$ is said to have *oracle*, or *black-box*, access to another algorithm $\mathcal{A}$ if $\mathcal{B}$ can invoke $\mathcal{A}$ on arbitrary inputs x and random coins r, which is denoted as $\mathcal{B}^{\mathcal{A}}$. The algorithm $\mathcal{B}$ is also said to be an *oracle algorithm*. If, for all inputs x, random coins r and algorithms $\mathcal{A}$, $\mathcal{B}^{\mathcal{A}}$ invokes $\mathcal{A}$ at most Q times, $\mathcal{B}$ is called a *Q-query* oracle algorithm.

2.3 Arithmetic Circuits

The main model of computation used in this dissertation is the arithmetic circuit model. Arithmetic circuits model the evaluation of multivariate polynomials $f(X_1, \ldots, X_n)$ defined over a finite field $\mathbb{F}$. They express a polynomial in terms of the basic arithmetic operations: addition and multiplication. More precisely, an arithmetic circuit is a directed acyclic graph. Its nodes are referred to as gates and its edges as wires. The gates with indegree 0 are called the input gates. Input gates have unbounded outdegree and are assigned a constant $a \in \mathbb{F}$ or a variable X_i. The remaining gates are addition or multiplication gates. They have indegree 2 and unbounded outdegree. As such, all wires naturally correspond to a multivariate polynomial in $\mathbb{F}[X_1, \ldots, X_n]$, where n is the number of variable input gates. An arithmetic circuit corresponding to the polynomial $f(X_1, \ldots, X_n)$ has a unique output gate with outdegree 0. Slightly abusing terminology, we also allow an arithmetic circuit C to have multiple output gates. In this case the circuit C corresponds to a vector of polynomials $(f_1, \ldots, f_s)$.

The evaluation of an arithmetic circuit entails assigning values to the n variables $X_1, \ldots, X_n$ and computing all wire values. For this reason, an arithmetic circuit with s output gates can also be viewed as a mapping $C\colon \mathbb{F}^n \to \mathbb{F}^s$.

The size $|C|$ of an arithmetic circuit C is the number of wires it contains. It is a measure for its computational complexity. There are many arithmetic circuits corresponding to the same function $f\colon \mathbb{F}^n \to \mathbb{F}^s$. A natural question is therefore to find the smallest arithmetic circuit computing a given function.

The *circuit satisfiability problem* asks to decide whether a given arithmetic circuit $C\colon \mathbb{F}^n \to \mathbb{F}$ admits a satisfiable input $\mathbf{x} \in \mathbb{F}^n$, i.e., an input $\mathbf{x}$ such that $C(\mathbf{x}) = 0$. The circuit satisfiability problem is NP-complete,[2] i.e., every problem in NP can be written as a circuit satisfiability problem, demonstrating its versatility.

2.4 Probability Distributions

Let us now recall some basic discrete probability theory. In this work, we will not require continuous probability theory.

[2] Recall that NP denotes the class of problems that admit an efficiently verifiable solution.

Definition 2.5 (Discrete Probability Space)**.** A discrete probability space is a tuple (Ω, p), containing a countable sample space Ω and a probability mass function $p\colon \Omega \to [0,1]$ such that $\sum_{\omega\in\Omega} p(\omega) = 1$. A subset $E \subseteq \Omega$ is called an event. Every event is associated to a probability via the probability measure

$$\Pr\colon 2^{\Omega} \to [0,1], \quad E \mapsto \sum_{\omega\in E} p(\omega)\,.$$

Definition 2.6 (Random Variable)**.** A random variable is a function $X\colon \Omega \to \mathcal{X}$ for some nonempty set $\mathcal{X}$. Moreover, the probability distribution of X is the function

$$D_X\colon \mathcal{X} \to [0,1], \quad x \mapsto \Pr(X = x) := \sum_{\omega\in X^{-1}(x)} p(\omega)\,.$$

For any $x \in \mathcal{X}$ and $C \subseteq \mathcal{X}$, the events $X^{-1}(x) \subseteq \Omega$ and $X^{-1}(C) \subseteq \Omega$ are simply denoted as $X = x$ and $X \in C$, respectively. The support of a random variable is $\operatorname{supp}(X) = \{x \in \mathcal{X} : \Pr(X = x) > 0\}$. Further, X is said to be uniformly distributed over $\mathcal{X}$ if $\mathcal{X}$ is a finite set and $\Pr(X = x) = 1/|\mathcal{X}|$ for all $x \in \mathcal{X}$. Sampling an element x from a distribution D_X is denoted as $x \leftarrow_R D_X$, i.e., $\Pr(x = y\colon x \leftarrow_R D_X) = \Pr(X = y)$ for all $y \in \mathcal{X}$. If D_X is the uniform distribution over some finite set $\mathcal{X}$, we also write $x \leftarrow_R \mathcal{X}$ instead of $x \leftarrow_R D_X$. For an algorithm $\mathcal{A}\colon \mathcal{X} \to \mathcal{Y}$, $\mathcal{A}(X)$ denotes the random variable with $\Pr(\mathcal{A}(X) = y) = \Pr\big(\mathcal{A}(x) = y : x \leftarrow_R D_X\big)$, where the probability is also over the randomness of $\mathcal{A}$.

Definition 2.7 (Statistical Distance)**.** The statistical distance between two random variables $X_0, X_1\colon \Omega \to \mathcal{X}$ is defined as

$$\Delta(X_0, X_1) = \frac{1}{2} \sum_{x\in\mathcal{X}} |\Pr(X_0 = x) - \Pr(X_1 = x)|\,.$$

The statistical distance is also called the *total variation distance*.

Towards proving the security of cryptographic protocols, we are often interested in algorithms $\mathcal{D}$ aiming to distinguish two random variables X_0 and X_1. For instance, the inability of an adversary to distinguish the encryption of a secret message from a uniformly random bitstring proves the security of the considered encryption scheme.

In order to quantify how well an algorithm $\mathcal{D}$ can distinguish two random variables X_0 and X_1 let us consider the following distinguishing game. First, a bit $b \leftarrow_R \{0,1\}$ is sampled uniformly at random. Second, an element $x \leftarrow_R D_{X_b}$ is sampled from the distribution of X_b. Finally, the algorithm $\mathcal{D}$, on input x, outputs a bit $b' \in \{0,1\}$. The algorithm $\mathcal{D}$ wins the distinguishing game if $b = b'$. For this reason, a probabilistic algorithm that always outputs a bit is also called a *distinguisher*.

The advantage $\operatorname{Adv}_{\mathcal{D}}(X_0, X_1)$ of a distinguisher now measures how well $\mathcal{D}$ succeeds in winning this game. For instance, the advantage equals 1 if the distinguisher always wins, and it equals 0 if $\mathcal{D}$ ignores the input x and outputs a random bit b', thereby always winning with probability 1/2.

Definition 2.8 (Advantage of a Distinguisher)**.** Let $X_0, X_1 \colon \Omega \to \mathcal{X}$ be two random variables and let $\mathcal{D} \colon \mathcal{X} \to \{0,1\}$ be a (probabilistic) distinguisher. Then, the advantage of $\mathcal{D}$ in distinguishing X_0 and X_1 is

$$\mathrm{Adv}_{\mathcal{D}}(X_0, X_1) := \left|\Pr\big(\mathcal{D}(X_0) = 0\big) - \Pr\big(\mathcal{D}(X_1) = 0\big)\right| .$$

Moreover, the advantage of a class of distinguishers $\mathcal{F}$ is

$$\mathrm{Adv}_{\mathcal{F}}(X_0, X_1) := \sup_{\mathcal{D} \in \mathcal{F}} \mathrm{Adv}_{\mathcal{D}}(X_0, X_1) .$$

The following lemma shows that the distinguishing advantage of a family of distinguishers is closely related to the statistical distance.

Lemma 2.1. *Let $X_0, X_1 \colon \Omega \to \mathcal{X}$ be random variables. Then*

$$\Delta(X_0, X_1) = \sup_{\mathcal{D}} \mathrm{Adv}_{\mathcal{D}}(X_0, X_1) ,$$

where the supremum is over all distinguishers $\mathcal{D}$.

Proof. See [CDN15, page 20]. □

We are now ready to define what it means for two families of random variables to be statistically or computationally indistinguishable.

Definition 2.9 (Statistical Indistinguishability)**.** Two families $\{X_s\}_{s \in S}$ and $\{Y_s\}_{s \in S}$ of random variables, indexed by a set of bitstrings $S \subseteq \{0,1\}^*$, are said to be statistically indistinguishable if the function

$$\Delta(s) := \Delta(X_s, Y_s)$$

is negligible in $|s|$. If $\Delta(s) = 0$ for all $s \in S$, $\{X_s\}_{s \in S}$ and $\{Y_s\}_{s \in S}$ are said to be perfectly indistinguishable.

Definition 2.10 (Computational Indistinguishability)**.** Let $\mathcal{F}$ be the class of polynomial time distinguishers. Two families $\{X_s\}_{s \in S}$ and $\{Y_s\}_{s \in S}$ of random variables, indexed by a set of bitstrings $S \subseteq \{0,1\}^*$, are said to be computationally indistinguishable if

$$\Delta(s) := \mathrm{Adv}_{\mathcal{F}}(X_s, X_s)$$

is negligible in $|s|$.

2.4.1 Geometric Distribution

A random variable B with two distinct possible outcomes, denoted 0 (failure) and 1 (success), is said to follow a Bernoulli distribution with parameter $p = \Pr(B = 1)$. Sampling from a Bernoulli distribution is also referred to as running a Bernoulli trial. The probability distribution of the number X of independent and identical Bernoulli trials needed to obtain a success is called the geometric distribution with parameter $p = \Pr(X = 1)$. In this case $\Pr(X = k) = (1-p)^{k-1}p$ for all $k \in \mathbb{N}$ and we write $X \sim \mathrm{Geo}(p)$. For two independent geometric distributions we have the following lemma.

Lemma 2.2. *Let $X \sim \mathrm{Geo}(p)$ and $Y \sim \mathrm{Geo}(q)$ be independently distributed. Then,*

$$\Pr(X \leq Y) = \frac{p}{p+q-pq} \geq \frac{p}{p+q}.$$

Proof. It holds that

$$\begin{aligned}\Pr(X \leq Y) &= \sum_{k=1}^{\infty} \Pr(X = k) \Pr(Y \geq k) = \sum_{k=1}^{\infty} (1-p)^{k-1} p \cdot (1-q)^{k-1} \\ &= p \sum_{\ell=0}^{\infty} (1-p)^{\ell} (1-q)^{\ell} = \frac{p}{1-(1-p)(1-q)} \\ &= \frac{p}{p+q-pq} \geq \frac{p}{p+q}\,,\end{aligned}$$

which completes the proof of the lemma. □

2.4.2 Negative Hypergeometric Distribution

Consider a bucket containing ℓ green balls and $N-\ell$ red balls, i.e., a total of N balls. In the negative hypergeometric experiment, balls are drawn uniformly at random from this bucket, without replacement, until k green balls have been found, or until the bucket is empty. The number of red balls X drawn in this experiment is said to have a *negative hypergeometric distribution* with parameters N, ℓ, k, which is denoted by $X \sim \mathrm{NHG}(N, \ell, k)$.

Lemma 2.3 (Negative Hypergeometric Distribution)**.** *Let $N, \ell, k \in \mathbb{N}$ with $\ell, k \leq N$, and let $X \sim \mathrm{NHG}(N, \ell, k)$. Then*

$$\mathbb{E}[X] \leq k \frac{N-\ell}{\ell+1}\,.$$

Proof. If $\ell < k$, it clearly holds that $\Pr(X = N-\ell) = 1$. Hence, in this case, $\mathbb{E}[X] = N - \ell \leq k\frac{N-\ell}{\ell+1}$, which proves the claim.

So let us now consider the case $\ell \geq k$. Then, for all $0 \leq x \leq N-\ell$,

$$\Pr(X = x) = \frac{\binom{x+k-1}{x}\binom{N-x-k}{N-\ell-x}}{\binom{N}{N-\ell}}\,.$$

Hence,

$$\begin{aligned}\mathbb{E}[X] &= \sum_{x=0}^{N-\ell} \Pr(X = x) \cdot x = \sum_{x=1}^{N-\ell} x \frac{\binom{x+k-1}{x}\binom{N-x-k}{N-\ell-x}}{\binom{N}{N-\ell}} \\ &= k\frac{N-\ell}{\ell+1} \sum_{x=1}^{N-\ell} \frac{\frac{x}{k}\binom{x+k-1}{x}\binom{N-x-k}{N-\ell-x}}{\frac{N-\ell}{\ell+1}\binom{N}{N-\ell}} = k\frac{N-\ell}{\ell+1} \sum_{x=1}^{N-\ell} \frac{\binom{x+k-1}{x-1}\binom{N-x-k}{N-\ell-x}}{\binom{N}{N-\ell-1}} \\ &= k\frac{N-\ell}{\ell+1} \sum_{x=1}^{N-\ell} \Pr(Y = x-1) = k\frac{N-\ell}{\ell+1}\,,\end{aligned}$$

where $Y \sim \mathrm{NHG}(N, \ell+1, k-1)$. This completes the proof of the lemma. □

Remark 2.1. Typically, negative hypergeometric experiments are restricted to the nontrivial case $\ell \geq k$. For reasons to become clear later, we also allow parameter choices with $\ell < k$ resulting in a trivial negative hypergeometric experiment in which all balls are always drawn.

Remark 2.2. The above negative hypergeometric experiment has a straightforward generalization to buckets with balls of more than 2 colors. Namely, say the bucket contains ℓ green balls and m_i balls of color i for $1 \leq i \leq M$. The experiment proceeds as before, i.e., drawing until either k green balls have been found or the bucket is empty. Let X_i be the number of balls of color i that are drawn in this experiment. Then $X_i \sim \mathrm{NHG}(\ell + m_i, \ell, k)$ for all i. To see this, simply run the generalized negative hypergeometric experiment without counting the balls that are neither green nor of color i.

2.5 Commitment Schemes

Commitment schemes allow a party, also referred to as a prover, to commit to (secret) input data. When a prover has made a commitment, the input data can no longer be changed, i.e., the commitment is *binding*. Moreover, the commitment itself does not reveal anything about the input data, i.e., it is *hiding*. Finally, at some later point in time, the prover can reveal his input data and prove that this was indeed the data it committed to before. This is called *opening* a commitment. Commitment schemes are one the most important building blocks in cryptography.

The following gives a formal definition for commitment schemes. The binding and hiding properties are not incorporated in this definition; we consider these as desirable security properties.

Definition 2.11 (Commitment Scheme)**.** A commitment scheme is defined by a probabilistic polynomial time setup algorithm Setup, which takes as input the (unary encoding of) a security parameter[3] λ and outputs a public key $\mathsf{pk} \leftarrow \textsc{Setup}(1^\lambda)$. Every public key defines a message set $\mathcal{M}_{\mathsf{pk}}$, a randomness set $\mathsf{Rand}_{\mathsf{pk}}$, a commitment set $\mathcal{C}_{\mathsf{pk}}$ and a deterministic function

$$\textsc{com}_{\mathsf{pk}} \colon \mathcal{M}_{\mathsf{pk}} \times \mathsf{Rand}_{\mathsf{pk}} \to \mathcal{C}_{\mathsf{pk}}, \quad (m;\gamma) \mapsto \textsc{com}_{\mathsf{pk}}(m;\gamma)\,.$$

To commit to a message $m \in \mathcal{M}_{\mathsf{pk}}$, a prover samples $\gamma \leftarrow_R \mathsf{Rand}_{\mathsf{pk}}$ uniformly at random and outputs the commitment $P = \textsc{com}_{\mathsf{pk}}(m;\gamma)$. A commitment is opened by revealing the message m together with the commitment randomness γ. An opening $(m;\gamma)$ of a commitment P is verified by checking that $\textsc{com}_{\mathsf{pk}}(m;\gamma) = P$. Let us now formally define what it means for a commitment scheme to be binding and hiding.

Definition 2.12 (Binding Commitment Scheme)**.** A commitment scheme defined by the setup algorithm Setup is (statistically) *binding* if, for every probabilistic

[3]The security parameter controls the expected amount of security a cryptographic primitive offers, i.e., there exists a monotone function f such that the cost of breaking the primitive instantiated with security parameter λ is at least $f(\lambda)$. Typically, we require the function f to grow faster than any polynomial $p(X) \in \mathbb{Z}[X]$.

algorithm $\mathcal{A}$,

$$\Pr\left(m_0 \neq m_1 \wedge P_0 = P_1 \;\middle|\; \begin{array}{c} \mathsf{pk} \leftarrow \textsc{Setup}\left(1^\lambda\right) \\ (m_0, \gamma_0, m_1, \gamma_1) \leftarrow \mathcal{A}(\mathsf{pk}) \\ P_0 = \textsc{com}_{\mathsf{pk}}(m_0; \gamma_0) \\ P_1 = \textsc{com}_{\mathsf{pk}}(m_1; \gamma_1) \end{array}\right) \leq \mathrm{negl}(\lambda)\,.$$

If the above probability equals 0, the commitment scheme is said to be *perfectly* binding. If the above only holds for polynomial time algorithms $\mathcal{A}$, the commitment scheme is said to be *computationally* binding.

Definition 2.13 (Hiding Commitment Scheme)**.** A commitment scheme defined by the setup algorithm SETUP is (statistically) *hiding* if, for every pair of probabilistic algorithms $(\mathcal{A}_1, \mathcal{A}_2)$,

$$\left|\Pr\left(\mathcal{A}_2(\mathsf{pk}, P) = b \;\middle|\; \begin{array}{c} \mathsf{pk} \leftarrow \textsc{Setup}\left(1^\lambda\right) \\ (m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{pk}) \\ b \leftarrow_R \{0,1\}, \gamma \leftarrow_R \mathsf{Rand}_{\mathsf{pk}} \\ P = \textsc{com}_{\mathsf{pk}}(m_b; \gamma) \end{array}\right) - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda)\,.$$

If the above probability equals $1/2$, the commitment scheme is said to be *perfectly* hiding. If the above only holds for polynomial time algorithm pairs $(\mathcal{A}_1, \mathcal{A}_2)$, the commitment scheme is said to be *computationally* hiding.

Note that if the commitment scheme is perfectly hiding, then $\textsc{com}(m;\gamma)$ and $\textsc{com}(m';\gamma)$ are identically distributed for all $m, m' \in \mathcal{M}_{\mathsf{pk}}$, where $\gamma \leftarrow_R \mathsf{Rand}_{\mathsf{pk}}$ is uniformly distributed.

A commitment scheme is said to be *homomorphic* if, for all public keys pk, the sets $\mathcal{M}_{\mathsf{pk}}$, $\mathsf{Rand}_{\mathsf{pk}}$ and $\mathcal{C}_{\mathsf{pk}}$ are groups, and the function $\textsc{com}_{\mathsf{pk}} \colon \mathcal{M}_{\mathsf{pk}} \times \mathsf{Rand}_{\mathsf{pk}} \to \mathcal{C}_{\mathsf{pk}}$ is a group homomorphism. Typically, the group operations in $\mathcal{M}_{\mathsf{pk}}$ and $\mathsf{Rand}_{\mathsf{pk}}$ are written additively and the group operation in $\mathcal{C}_{\mathsf{pk}}$ is written multiplicatively.

We say that a commitment scheme is a *vector* commitment scheme if the setup algorithm additionally takes as input a dimension n and, for every public key $\mathsf{pk} \leftarrow \textsc{Setup}(1^\lambda, n)$, the message set is an n-fold Cartesian product $\mathcal{M}_{\mathsf{pk}}^n$, i.e.,

$$\textsc{com}_{\mathsf{pk}} \colon \mathcal{M}_{\mathsf{pk}}^n \times \mathsf{Rand}_{\mathsf{pk}} \to \mathcal{C}_{\mathsf{pk}}\,.$$

A vector commitment scheme thus allows a prover to commit to vectors of arbitrary length n. If the commitment scheme is homomorphic and $n' < n$, we also write $\textsc{com}_{\mathsf{pk}}(m_1, \ldots, m_{n'}; \gamma) := \textsc{com}_{\mathsf{pk}}(m_1, \ldots, m_{n'}, 0, \ldots, 0; \gamma)$ where $(m_1, \ldots, m_{n'}, 0, \ldots, 0) \in \mathcal{M}_{\mathsf{pk}}^n$. Sometimes, if $n' > n$, we abuse notation and still write $\textsc{com}_{\mathsf{pk}}(m_1, \ldots, m_{n'}; \gamma)$. In this case, we implicitly assume that the commitment scheme was actually instantiated with dimension at least n'.

A vector commitment scheme is said to be *compact* if the size of a commitment is constant in n. Moreover, it is said to be *compressing* if the size of a commitment is sublinear in n, i.e., the size of a commitment grows sublinearly in the dimension n of the committed vector. In particular, any compact vector commitment scheme is compressing. It is easily seen that a compressing commitment scheme can be at most computationally binding.

2.6 Group-Based Cryptographic Assumptions

The security of many cryptographic protocols is based on the intractability of certain computational problems. In this section, we introduce and formalize the group-based cryptographic hardness assumptions that are used in this dissertation.

One of the best-known computational problems used in cryptography is the *discrete logarithm* (DL) problem. Let $(\mathbb{G}, \cdot)$ be a group of prime order q and let $g \neq 1$. Then g generates $\mathbb{G}$, i.e., for all $h \in \mathbb{G}$ there exists an $x \in \mathbb{Z}_q$ such that $g^x = h$. The exponent x is also called the discrete logarithm of h with respect to generator g. The DL problem asks to find x given g and h. In suitable groups, this problem is assumed to be intractable, i.e., polynomial-time algorithms succeed with at most negligible probability in solving this problem. The following definition formalizes the discrete logarithm assumption.

Definition 2.14 (Discrete Logarithm Assumption)**.** Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that, on input a security parameter λ, outputs a prime q, a group $(\mathbb{G}, \cdot)$ of order q and a generator g of $\mathbb{G}$. The *discrete logarithm* (DL) assumption holds for $\mathcal{G}$ if for all probabilistic polynomial time algorithms $\mathcal{A}$

$$\Pr\big(h = g^x : (q, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda) \wedge h \leftarrow_R \mathbb{G} \wedge x \leftarrow \mathcal{A}(q, \mathbb{G}, g, h)\big) \leq \mathrm{negl}(\lambda)\,.$$

The second group based hardness assumption is the *decisional Diffie-Hellman* (DDH) assumption [Bon98]. This assumption states that it is hard for an adversary to distinguish triples of the form (g^x, g^y, g^{xy}) from those of the form (g^x, g^y, g^z), where $x, y, z \leftarrow_R \mathbb{Z}_q$ are sampled uniformly at random. The DL assumption is implied by the DDH assumption, i.e., if the DDH assumption holds, so does the DL assumption.

Definition 2.15 (Decisional Diffie-Hellman Assumption)**.** Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that, on input a security parameter λ, outputs a prime q, a group $(\mathbb{G}, \cdot)$ of order q and a generator g of $\mathbb{G}$. The *decisional Diffie-Hellman* (DDH) assumption holds for $\mathcal{G}$ if for all probabilistic polynomial time algorithms $\mathcal{A}$

$$\big|\Pr\big(\mathcal{A}(q, \mathbb{G}, g, g^x, g^y, g^{xy}) = 1\big) - \Pr\big(\mathcal{A}(q, \mathbb{G}, g, g^x, g^y, g^z) = 1\big)\big| \leq \mathrm{negl}(\lambda)\,,$$

where the probabilities are over $(q, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda)$, $x, y, z \leftarrow_R \mathbb{Z}_q$ and $\mathcal{A}$'s randomness.

We also refer to the algorithm $\mathcal{G}$ in definitions 2.14 and 2.15 as a prime order group generator. In some settings, the algorithm $\mathcal{G}$ actually outputs a bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, e)$. In this case, we must specify in which of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{H}$ the DL or DDH assumption holds. In particular, if the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$, we say that the *symmetrical external Diffie-Hellman* (SXDH) assumption [BGM+05] holds. It is easily seen that, for a bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, e)$, the existence of an efficiently computable isomorphism $\psi\colon \mathbb{G}_1 \to \mathbb{G}_2$ contradicts the DDH assumption in $\mathbb{G}_1$, and vice-versa the existence of an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ contradicts the DDH assumption in $\mathbb{G}_2$. Hence, the SXDH assumption can only hold if there do not exist efficiently

computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. This class of bilinear groups (or pairings) is also referred to as Type III [GPS08].

The product N of two primes p and q is called an *RSA-modulus*. It is assumed to be hard to find the prime factors p and q of N. Further, the group $\mathbb{Z}_N^*$ of multiplicative units modulo N, also referred to as an *RSA-group*, has cardinality $\phi(N) = (p-1)(q-1)$. From this it follows that, without knowledge of p and q, it is intractable to find the order of the group $\mathbb{Z}_N^*$; if not, one could efficiently factor RSA-moduli. For this reason, the group $\mathbb{Z}_N^*$ is also said to be of *hidden order*.

There exists a broad variety of hardness assumptions based on groups with hidden order; we introduce two of them. First, the *strong-RSA* assumption [BP97] states that it is hard to compute nontrivial roots in a group $\mathbb{G}$ with hidden order. Second, the *hidden order* assumption states that it is hard to find the order of group elements $g \leftarrow_R \mathbb{G}$ sampled uniformly at random. The hidden order assumption is implied by the strong-RSA assumption.

A disadvantage of RSA-groups is that their order is only hidden from parties that are oblivious to the prime factors p and q of N. In practice, this means that the RSA-group typically has to be generated by a trusted dealer. An alternative candidate for groups of hidden order are class groups of imaginary quadratic number fields [Wes19; BFS20; BHR+21]. Class groups can be generated in a transparent manner and thus do not require a trusted dealer.

Definition 2.16 (Strong-RSA Assumption)**.** Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that, on input a security parameter λ, outputs a group $(\mathbb{G}, \cdot)$ (with hidden order). The *strong-RSA* assumption holds for $\mathcal{G}$ if for all probabilistic polynomial time algorithms $\mathcal{A}$,

$$\Pr\big(g = P^x \wedge x > 1 : \mathbb{G} \leftarrow \mathcal{G}(1^\lambda) \wedge g \leftarrow_R \mathbb{G} \wedge (P, x) \leftarrow \mathcal{A}(\mathbb{G}, g)\big) \leq \mathrm{negl}(\lambda)\,.$$

Definition 2.17 (Hidden Order Assumption)**.** Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that, on input a security parameter λ, outputs a group $(\mathbb{G}, \cdot)$ (with hidden order). The *hidden order* assumption holds for $\mathcal{G}$ if for all probabilistic polynomial time algorithms $\mathcal{A}$,

$$\Pr\big(g^x = 1 \wedge x > 1 : \mathbb{G} \leftarrow \mathcal{G}(1^\lambda) \wedge g \leftarrow_R \mathbb{G} \wedge x \leftarrow \mathcal{A}(\mathbb{G}, g)\big) \leq \mathrm{negl}(\lambda)\,.$$

2.7 Lattices and Lattice Problems

A disadvantage of the group-based assumptions of the previous section is that, once available, a quantum computer will be able to solve the corresponding computational problems efficiently [Sho94]. Therefore, cryptographic primitives based on these assumptions will in general not be secure against adversaries with access to a quantum computer. By contrast, *post-quantum cryptography* studies the design of cryptographic primitives based on computational problems that are intractable even for quantum adversaries. One of the most promising areas in this field of research is *lattice-based* cryptography, where the underlying problems are so-called lattice problems. In this section, we introduce a number of variants of the short integer solution (SIS) problem.

A lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$. The lattice Λ is said to be q-ary if $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$. For instance, for any $A \in \mathbb{Z}_q^{k\times m}$ the sets

$$\begin{aligned}\Lambda_q(A) &= \{\mathbf{y} \in \mathbb{Z}^k : \exists \mathbf{x} \in \mathbb{Z}^m \ A\mathbf{x} = \mathbf{y} \mod q\} \quad \text{and}\\ \Lambda_q^\perp(A) &= \{\mathbf{x} \in \mathbb{Z}^m : A\mathbf{x} = 0 \mod q\}\end{aligned}$$

are q-ary lattices in $\mathbb{Z}^k$ and $\mathbb{Z}^m$ respectively. Finding a nonzero and "short" element in the lattice $\Lambda_q^\perp(A) \subseteq \mathbb{Z}^m$ is referred to as the *Short Integer Solution* (SIS) problem [Ajt96].

Definition 2.18 ($\mathrm{SIS}_{q,k,m,\beta}$-Problem [Ajt96])**.** The $\mathrm{SIS}_{q,k,m,\beta}$-problem is defined as follows: Given a matrix $A \leftarrow_R \mathbb{Z}_q^{k\times m}$ sampled uniformly at random, find a nonzero vector $\mathbf{s} \in \mathbb{Z}^m$, such that $A\mathbf{s} = 0 \mod q$ and $\|\mathbf{s}\|_2 \leq \beta$.

Let $\mathcal{R} = \mathbb{Z}[X]/f(X)$ for a monic[4] polynomial $f(X)$ of degree d. The coefficient embedding

$$\psi\colon \mathcal{R} \to \mathbb{Z}^d, \quad \sum_{i=1}^{d} a_i X^{i-1} \mapsto (a_1, \ldots, a_d)$$

is a group isomorphism. Hence, $\mathcal{R}$ corresponds to the lattice $\mathbb{Z}^d$. Moreover, every ideal $I \subseteq \mathcal{R}$ corresponds to a sublattice $\psi(I) \subseteq \mathbb{Z}^d$. The lattice $\psi(I)$ is said to be a *structured* or *ideal* lattice.

For $q \in \mathbb{N}$, we write $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/\big(f(X)\big)$. Further, to $a_1, \ldots, a_m \in \mathcal{R}_q$, we associate the following q-ary lattice

$$\Lambda_q^\perp(a_1, \ldots, a_m) = \{\mathbf{x} \in \mathcal{R}^m : \textstyle\sum_{i=1}^m a_i x_i = 0 \mod q\}.$$

The coefficient embedding ψ also equips the rings $\mathcal{R}$ and $\mathcal{R}^m$ with a geometry. More precisely, we define $\|\mathbf{x}\| = \|\psi(\mathbf{x})\|$ for any $\mathbf{x} \in \mathcal{R}^m$ and any norm $\|\cdot\|$ on $\mathbb{Z}^{dm}$. Finding a nonzero and short element in the lattice $\Lambda_q^\perp(a_1, \ldots, a_m) \subseteq \mathcal{R}^m$ is referred to as the *Ring-SIS* (RSIS) problem [PR06; LM06].

Definition 2.19 ($\mathrm{RSIS}_{q,m,\beta}$-Problem [Ajt96])**.** Let $\mathcal{R} = \mathbb{Z}[X]/f(X)$ for a monic polynomial $f(X)$. The $\mathrm{RSIS}_{q,m,\beta}$-problem over $\mathcal{R}$ is defined as follows: Given $a_1, \ldots, a_m \leftarrow_R \mathcal{R}_q$ sampled uniformly at random, find a nonzero vector $\mathbf{s} = (s_1, \ldots, s_m) \in \mathcal{R}^m$, such that $\sum_{i=1}^m a_i s_i = 0 \mod q$ and $\|\mathbf{s}\|_2 \leq \beta$.

For $A \in \mathcal{R}_q^{k\times m}$, $\Lambda_q^\perp(A) = \{\mathbf{x} \in \mathcal{R}^m : A\mathbf{x} = 0 \mod q\}$ corresponds to a q-ary sublattice of $\mathbb{Z}^{dm}$. The set $\Lambda_q^\perp(A) \subseteq \mathcal{R}^m$ is a finitely generated $\mathcal{R}$-module. For this reason, the corresponding lattice is also called a *module lattice*. Finding a nonzero and short element in a lattice $\Lambda_q^\perp(A)$, for $A \in \mathcal{R}_q^{k\times m}$, is referred to as the *Module-SIS* (MSIS) problem [LS15]. The MSIS-problem is a generalization of both the SIS- and the RSIS-problem. It is assumed to be intractable, even for quantum computers.

Definition 2.20 ($\mathrm{MSIS}_{q,k,m,\beta}$-Problem [LS15])**.** Let $\mathcal{R} = \mathbb{Z}[X]/f(X)$ for a monic polynomial $f(X)$. The $\mathrm{MSIS}_{q,k,m,\beta}$-problem over $\mathcal{R}$ is defined as follows: Given a matrix $A \leftarrow_R \mathcal{R}_q^{k\times m}$ sampled uniformly at random, find a nonzero vector $\mathbf{s} \in \mathcal{R}^m$, such that $A\mathbf{s} = 0 \mod q$ and $\|\mathbf{s}\|_2 \leq \beta$.

[4] Recall that a polynomial $f(X) = \sum_{i=0}^n a_i X^n$ is said to be monic if its leading coefficient a_n equals 1.

The *Gaussian heuristic* states that the length $\lambda_1\big(\Lambda_q^\perp(A)\big) = \|\mathbf{s}\|_2 \in \mathbb{R}_{\geq 0}$ of the shortest vector $\mathbf{s}$ of a q-ary lattice $\Lambda_q^\perp(A)$, for $A \in \mathcal{R}_q^{k\times m}$, is approximately equal to $\sqrt{m/(2\pi e)}q^{k/m}$ [MR09]. The quality of an algorithm χ for finding short vectors in a lattice can be characterized by its root Hermite factor δ, which is defined such that χ is expected to output basis vectors $\mathbf{s}$ with

$$\|\mathbf{s}\|_2 \approx \min(q, \delta^{dm} q^{k/m})\,. \tag{2.1}$$

In particular, smaller values of δ require better algorithms or a longer runtime. Given the current state-of-the-art, a (quantum) algorithm with $\delta \approx 1.0045$ is assumed to take at least 2^{128} operations [APS15; ESS+19], i.e., $\delta \approx 1.0045$ plausibly provides 128-bit post-quantum security.

Micciancio and Regev [MR09] showed that, from Equation 2.1, it follows that it is often suboptimal to apply the algorithm χ directly to the lattice of interest. For simplicity, let us consider the SIS-problem, i.e., we consider a lattice $\Lambda_q(A)$ with $A \in \mathbb{Z}_q^{k\times m}$, and aim to find a short vector in $\Lambda_q(A)$. For large enough m, the algorithm χ should be applied to a related lattice in $\Lambda_q(A') \subseteq \mathbb{Z}^{m'}$ with

$$m' = \sqrt{\frac{k\log_2(q)}{\log_2(\delta)}}\,.$$

More precisely, if $m > m'$, let $A' \in \mathbb{Z}_q^{k\times m'}$ be a submatrix of A obtained by removing $m - m'$ columns of A. The short vector output by χ applied to $\Lambda_q(A')$ can be appended with $m-m'$ zeros to obtain an element of $\Lambda_q(A)$ with exactly the same norm. Interestingly, for a fixed root Hermite factor δ, this approach outputs shorter vectors than applying χ directly to $\Lambda_q(A)$. In fact, the above approach is expected to output vectors of length

$$\|\mathbf{s}\|_2 \geq \min\Big(q, 2^{2\sqrt{k\log\delta\log q}}\Big)\,.$$

Note that this norm-bound is independent of the dimension m. Hence, when m is large enough, the parameter m does not influence the hardness of the SIS-problem. The same approach applied to the MSIS-problems, where $A \in \mathcal{R}_q^{k\times m}$, is expected to output lattice elements $\mathbf{s} \in \Lambda_q(A) \subseteq \mathcal{R}_q^m$ of norm

$$\|\mathbf{s}\|_2 \geq \min\Big(q, 2^{2\sqrt{dk\log\delta\log q}}\Big)\,, \tag{2.2}$$

where d is the degree the ring extension $\mathcal{R} = \mathbb{Z}[X]/f(X)$ over $\mathbb{Z}$.

In this work, we will mainly be interested in vectors that are short with respect to the ℓ_∞-norm. For this reason we also consider the following variant of the MSIS-problem, where "shortness" is defined in terms of the ℓ_∞-norm. Clearly, the hardness of $\mathrm{MSIS}^\infty_{q,k,m,\beta}$ is implied by the hardness of $\mathrm{MSIS}_{q,k,m,\sqrt{dm}\beta}$.

Definition 2.21 ($\mathrm{MSIS}^\infty_{q,k,m,\beta}$ Problem)**.** Let $\mathcal{R} = \mathbb{Z}[X]/f(X)$ for a monic polynomial $f(X)$. The $\mathrm{MSIS}^\infty_{q,k,m,\beta}$ problem over $\mathcal{R}$ is defined as follows: Given a matrix $A \leftarrow_R \mathcal{R}_q^{k\times m}$ sampled uniformly at random, find a nonzero vector $\mathbf{s} \in \mathcal{R}^m$ such that $A\mathbf{s} = 0 \mod q$ and $\|\mathbf{s}\|_\infty \leq \beta$.

2.8 Interactive (Zero-Knowledge) Proofs

A *binary relation* $\mathfrak{R}$ is a subset of the Cartesian product $X \times Y$ of two sets X and Y. It describes a connection between elements of X and elements of Y. Unless stated otherwise, we assume X and Y to be the set of arbitrary length bit strings $\{0,1\}^*$, and thus relations $\mathfrak{R}$ to be subsets of $\{0,1\}^* \times \{0,1\}^*$.

Following standard terminology, a string $w \in \{0,1\}^*$ is called a *witness* for the *statement* $x \in \{0,1\}^*$ if $(x;w) \in \mathfrak{R}$. The set of valid witnesses for a statement x is denoted by $\mathfrak{R}(x)$, i.e., $\mathfrak{R}(x) = \{w : (x;w) \in \mathfrak{R}\}$. A statement that admits a witness is said to be a *true* or *valid* statement. The set of true statements is denoted by $L_{\mathfrak{R}}$, i.e., $L_{\mathfrak{R}} = \{x : \exists\, w \text{ s.t. } (x;w) \in \mathfrak{R}\}$. A binary relation is said to be an NP relation if the validity of a witness w can be verified in time polynomial in the size $|x|$ of the statement x. In particular, for an NP relation, it holds that the size $|w|$ of a witness $w \in \mathfrak{R}(x)$ is polynomial in $|x|$. From now on we assume all relations to be NP relations.

An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ aims for a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a statement x admits a witness, or even that the prover *knows* a witness $w \in \mathfrak{R}(x)$.

Definition 2.22 (Interactive Proof)**.** An *interactive proof* $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is an interactive protocol between two probabilistic machines, a prover $\mathcal{P}$ and a polynomial time verifier $\mathcal{V}$. Both $\mathcal{P}$ and $\mathcal{V}$ take as public input a statement $x \in \{0,1\}^*$, and additionally, $\mathcal{P}$ takes as private input a witness $w \in \mathfrak{R}(x)$, which is denoted as $\Pi(x;w)$ or $(\mathcal{P}(w), \mathcal{V})(x)$. As the output of the protocol, $\mathcal{V}$ either accepts or rejects the statement. Accordingly, we say the corresponding transcript (i.e., the set of all messages exchanged in the protocol execution) is *accepting* or *rejecting*.

An interactive proof Π is *complete* if the verifier $\mathcal{V}$ accepts honest executions with a public-private input pair $(x;w) \in \mathfrak{R}$ with large probability, i.e., the claims made by honest provers are accepted with large probability. It is *sound* if the verifier rejects false statements $x \notin L_{\mathfrak{R}}$ with large probability, i.e., the claims made by dishonest provers are rejected with large probability. Originally interactive proofs were defined to be complete and sound [GMR85]. By contrast, we do not require interactive protocols to satisfy these properties by definition, but consider them as desirable security properties.

Definition 2.23 (Completeness)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *complete* with completeness error $\rho\colon \mathbb{N} \to [0,1]$ if for all $(x;w) \in \mathfrak{R}$,

$$\Pr\big((\mathcal{P}(w), \mathcal{V})(x) = \mathsf{reject}\big) \leq \rho(|x|)\,.$$

If $\rho(|x|) = 0$ for all x, $(\mathcal{P}, \mathcal{V})$ is said to be perfectly complete.

Definition 2.24 (Soundness)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *sound* with soundness error $\sigma\colon \mathbb{N} \to [0,1]$ if for all $x \notin L_{\mathfrak{R}}$ and every prover $\mathcal{P}^*$,

$$\Pr\big((\mathcal{P}^*, \mathcal{V})(x) = \mathsf{accept}\big) \leq \sigma(|x|)\,.$$

If this property only holds for (probabilistic) polynomial time (i.e., computationally bounded) provers $\mathcal{P}^*$, then Π is said to be *computationally* sound.

Let us consider some additional (desirable) properties of interactive proofs. We assume that the prover $\mathcal{P}$ sends the first and the last message in any interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$. If this is not the case, the interactive proof can be appended with an empty message. Hence, the number of communication rounds $2\mu + 1$ is always odd. We also say Π is a $(2\mu + 1)$-round protocol. We will refer to *multi-round* protocols as a way of emphasizing that we are not restricting to 3-round protocols.

Definition 2.25 (Public-Coin)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ is *public-coin* if all of $\mathcal{V}$'s random choices are made public.

If a protocol is public-coin, the verifier only needs to send its random choices to the prover. In this case, $\mathcal{V}$'s messages are also referred to as *challenges*, and the set from which $\mathcal{V}$ samples its messages uniformly at random is called the challenge set.

We refer to a 3-round public-coin interactive proof as a Σ-*protocol*. Note that often a Σ-protocol is required to be (perfectly) complete, special-sound and special honest-verifier zero-knowledge (SHVZK) by definition. However, we do not require a Σ-protocol to have these additional properties.

Definition 2.26 (Σ-Protocol)**.** A Σ-protocol is a 3-round public-coin interactive proof.

2.8.1 Knowledge Soundness

If an interactive proof is complete and sound, it "merely" allows a prover to convince a verifier that a statement x admits a witness, i.e., $x \in L_{\mathfrak{R}}$. It does not necessarily convince a verifier that the prover "knows" a witness $w \in \mathfrak{R}(x)$. Informally, a prover $\mathcal{P}^*$ is said to know a witness w if it can *compute* this witness efficiently. More precisely, knowledge of w requires the existence of an efficient algorithm that, given x and *oracle* access to $\mathcal{P}^*$, outputs a witness $w \in \mathfrak{R}(x)$. For a more elaborate discussion on the definition of knowledge we refer to [Gol04].

The above allows us to define what it means for an interactive proof to prove knowledge of a witness w. This stronger notion of soundness is called *knowledge soundness* and is formally defined in Definition 2.27.

Definition 2.27 (Knowledge Soundness)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *knowledge sound* with knowledge error $\kappa \colon \mathbb{N} \to [0, 1]$ if there exists a positive polynomial q and an algorithm $\mathcal{E}$, called a *knowledge extractor*, with the following properties: The extractor $\mathcal{E}^{\mathcal{P}^*}(x)$, given input x and oracle access to a (potentially dishonest) prover $\mathcal{P}^*$, runs in an expected number of steps that is polynomial in $|x|$ and outputs a witness $w \in \mathfrak{R}(x)$ with probability

$$\Pr\big((x; \mathcal{E}^{\mathcal{P}^*}(x)) \in \mathfrak{R}\big) \geq \frac{\epsilon(x, \mathcal{P}^*) - \kappa(|x|)}{q(|x|)},$$

where $\epsilon(x, \mathcal{P}^*) := \Pr\big((\mathcal{P}^*, \mathcal{V})(x) = \mathsf{accept}\big)$.

If these properties only hold for probabilistic polynomial time (i.e., computationally bounded) provers $\mathcal{P}^*$, then Π is said to be *computationally* knowledge sound.

The extraction algorithm of Definition 2.27 only has oracle or black-box access to $\mathcal{P}^*$. For this reason, this is also referred to as *black-box extraction*. Moreover, the efficiency of an extractor is oftentimes measured in the (expected) number of times it invokes, or queries, $\mathcal{P}^*$.

If $\epsilon(x, \mathcal{P}^*) = \Pr\big((\mathcal{P}^*, \mathcal{V})(x) = \mathsf{accept}\big) > \kappa(|x|)$, then the success probability of the knowledge extractor of Definition 2.27 is positive. Hence, $\epsilon(x, \mathcal{P}^*) > \kappa(|x|)$ implies that x admits a witness, i.e., $x \in L_{\mathfrak{R}}$. It therefore follows that knowledge soundness with knowledge error $\kappa(|x|)$ implies soundness with soundness error $\sigma(|x|) = \kappa(|x|)$. Hence, knowledge soundness is indeed a stronger property than soundness.

Remark 2.3. It is straightforward to verify that, in order to satisfy Definition 2.27, it is sufficient to show that the required property holds for *deterministic* provers $\mathcal{P}^*$. Namely, let $\mathcal{P}^*$ be an arbitrary probabilistic dishonest prover, and let $\mathcal{P}^*[r]$ be the deterministic prover obtained by fixing $\mathcal{P}^*$'s randomness to r. Then $\epsilon(x, \mathcal{P}^*) = \mathbb{E}_r[\epsilon(x, \mathcal{P}^*[r])]$, where $\mathbb{E}_r$ denotes the expectation over the random choice of r. Furthermore, if $\mathcal{E}^{\mathcal{P}^*}(x)$ is declared to run $\mathcal{E}^{\mathcal{P}^*[r]}(x)$ for a random choice of r, then the same holds for the success probability of the extractor:

$$\Pr\big((x; \mathcal{E}^{\mathcal{P}^*}(x)) \in \mathfrak{R}\big) = \mathbb{E}_r\big[\Pr\big((x; \mathcal{E}^{\mathcal{P}^*[r]}(x)) \in \mathfrak{R}\big)\big] .$$

It follows that in order to satisfy Definition 2.27, it is sufficient to show that the required property holds for *deterministic* provers $\mathcal{P}^*$. For this reason, we may assume provers to be deterministic, in particular, we will consider the prover's first message to be deterministic. This will significantly simplify our analysis.

Definition 2.27 deviates from the more common textbook definition of knowledge soundness [Gol04; HL10] given in Definition 2.28. Instead of requiring the existence of an extractor that runs in expected polynomial time and succeeds with probability at least $(\epsilon(x, \mathcal{P}^*) - \kappa(|x|))/q(|x|)$, the textbook definition requires the existence of an extractor that, as long as $\epsilon(x, \mathcal{P}^*) > \kappa(|x|)$, *always* succeeds, but has an expected runtime that is inversely proportional to $\epsilon(x, \mathcal{P}^*) - \kappa(|x|)$. In particular, the latter extractor does not necessarily run in polynomial time. The two definitions are known to be equivalent [Gol04, Proposition 4.7.4] and therefore display a trade-off between the success probability and the expected runtime of the extractor. We will be using Definition 2.27, since this formulation simplifies our analysis. It is, for instance, much less obvious that it is sufficient to consider only deterministic provers if one uses Definition 2.28 directly.

Definition 2.28 (Knowledge Soundness - Equivalent Definition)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *knowledge sound* with knowledge error $\kappa\colon \mathbb{N} \to [0, 1]$ if there exists a positive polynomial q and an algorithm $\mathcal{E}$, called a *knowledge extractor*, with the following properties: The extractor $\mathcal{E}^{\mathcal{P}^*}(x)$, given input x and oracle access to a (potentially dishonest) prover $\mathcal{P}^*$ with $\epsilon(x, \mathcal{P}^*) := \Pr\big((\mathcal{P}^*, \mathcal{V})(x) = \mathsf{accept}\big) > \kappa(|x|)$, outputs a witness $w \in \mathfrak{R}(x)$ in an expected number of steps bounded by

$$\frac{q(|x|)}{\epsilon(x, \mathcal{P}^*) - \kappa(|x|)} .$$

Remark 2.4. By Definition 2.28 it is obvious that, in order to prove knowledge soundness, it is enough to consider statements $x \in \{0,1\}^*$ for which the prover $\mathcal{P}^*$ succeeds with probability $\epsilon(x, \mathcal{P}^*) > \kappa(|x|)$, i.e., there are no requirements on the behavior of the extractor for statements x with $\epsilon(x, \mathcal{P}^*) \leq \kappa(|x|)$. By contrast, Definition 2.27 requires extractors to be efficient for *all* statements x. This seems to be a stronger requirement, however the equivalence between these two definitions proves the contrary. Therefore, also towards satisfying Definition 2.27, it is enough to consider statements x with $\epsilon(x, \mathcal{P}^*) > \kappa(|x|)$. Since almost all our knowledge extractors are efficient for all x, we typically do not have to distinguish between statements x with $\epsilon(x, \mathcal{P}^*) > \kappa(|x|)$ and statements x with $\epsilon(x, \mathcal{P}^*) \leq \kappa(|x|)$.

Remark 2.5. In principle one could allow the completeness, soundness and knowledge error to be functions of the statement x instead of its size $|x|$. Both versions appear in literature, e.g., Goldreich [Gol04] defines these errors as functions of $|x|$, whereas Hazay and Lindell [HL10] define them as functions of x.

Remark 2.6. Sometimes a slightly weaker definition of knowledge soundness is used [BG92; Gol04; HL10]. This weaker definition decouples knowledge soundness from soundness by only requiring the extractor to run in expected polynomial time on inputs $x \in L_{\mathfrak{R}}$, i.e., it does not require the protocol to be sound. The reason is that in some applications the public input is guaranteed to be a *true* statement, i.e., admitting a witness. In these applications it does not matter how the protocol behaves on inputs $x \notin L_{\mathfrak{R}}$, i.e., the protocol does not need to be sound. It is straightforward to show that a *sound* protocol satisfying this weaker notion of knowledge soundness is also knowledge sound in the stronger sense of Definition 2.27.

Definition 2.29 (Proof of Knowledge)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ that is both complete with completeness error $\rho(\cdot)$ and knowledge sound with knowledge error $\kappa(\cdot)$ is a *Proof of Knowledge* (PoK) if there exists a polynomial q such that $1 - \rho(|x|) \geq \kappa(|x|) + 1/q(|x|)$ for all x.

Definition 2.30 (Argument of Knowledge)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ that is both complete with completeness error $\rho(\cdot)$ and *computationally* knowledge sound with knowledge error $\kappa(\cdot)$ is an *Argument of Knowledge* (AoK) if there exists a polynomial q such that $1 - \rho(|x|) \geq \kappa(|x|) + 1/q(|x|)$ for all x.

Sometimes the alternative, nonequivalent, notion of knowledge soundness presented in Definition 2.31 is used [Cra96; HM98; Unr12]. In this alternative notion, the knowledge extractor is required to run in *strict* polynomial time instead of *expected* polynomial time. However, its success probability is allowed to be proportional to $\big(\epsilon(x, \mathcal{P}^*) - \kappa(|x|)\big)^c$ for an arbitrary constant $c \geq 1$, whereas Definition 2.27 requires the success probability of the extractor to be proportional to $\epsilon(x, \mathcal{P}^*) - \kappa(|x|)$. For some interactive proofs this degradation of the success probability indeed allows the construction of *strict*, instead of *expected*, polynomial time knowledge extractors. Note that, since the success probability of the extractor degrades exponentially in c, this alternative definition only gives a meaningful notion of knowledge soundness if the exponent c is indeed constant.

Definition 2.31 (Knowledge Soundness - Alternative Notion)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is said to satisfy the alternative notion of *knowledge soundness* with knowledge error $\kappa\colon \mathbb{N} \to [0,1]$ if there exists a positive polynomial q, a constant $c \geq 1$ and an algorithm $\mathcal{E}$, called a *knowledge extractor*, with the following properties: The extractor $\mathcal{E}^{\mathcal{P}^*}(x)$, given input x and oracle access to a (potentially dishonest) prover $\mathcal{P}^*$, runs in an expected number of steps that is polynomial in $|x|$ and, if $\epsilon(x, \mathcal{P}^*) > \kappa(|x|)$, outputs a witness $w \in \mathfrak{R}(x)$ with probability

$$\Pr\big((x; \mathcal{E}^{\mathcal{P}^*}(x)) \in \mathfrak{R}\big) \geq \frac{\big(\epsilon(x, \mathcal{P}^*) - \kappa(|x|)\big)^c}{q(|x|)},$$

where $\epsilon(x, \mathcal{P}^*) := \Pr\big((\mathcal{P}^*, \mathcal{V})(x) = \mathsf{accept}\big)$.

2.8.2 Special-Soundness

We recall the notion of (general) *special-soundness*. It is typically easier to prove that an interactive proof is special-sound than to prove that it is knowledge sound. Note that we require special-sound protocols to be public-coin.

Definition 2.32 (k-out-of-N Special-Soundness)**.** Let $k, N \in \mathbb{N}$. A 3-round public-coin interactive proof Π for relation $\mathfrak{R}$, with challenge set of cardinality $N \geq k$, is *k-out-of-N special-sound* if there exists a polynomial time algorithm that, on input a statement x and k accepting transcripts $(a, c_1, z_1), \ldots (a, c_k, z_k)$ with common first message a and pairwise distinct challenges $c_1, \ldots, c_k$, outputs a witness $w \in \mathfrak{R}(x)$. We also say Π is k-special-sound and, if $k = 2$, it is simply said to be special-sound.

In order to generalize k-special-soundness to multi-round protocols, we introduce the notion of a tree of transcripts.

Definition 2.33 (Tree of Transcripts)**.** Let $\mathbf{k} = (k_1, \ldots, k_\mu) \in \mathbb{N}^\mu$. A $\mathbf{k}$-tree of transcripts for a $(2\mu + 1)$-round public-coin interactive proof Π is a set of $K = \prod_{i=1}^{\mu} k_i$ transcripts arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf node. For a graphical representation we refer to Figure 2.1. We refer to the corresponding tree of challenges as a $\mathbf{k}$-tree of challenges.

Definition 2.34 ($\mathbf{k}$-out-of-$\mathbf{N}$ Special-Soundness)**.** Let $\mathbf{k} = (k_1, \ldots, k_\mu)$, $\mathbf{N} = (N_1, \ldots, N_\mu) \in \mathbb{N}^\mu$. A $(2\mu + 1)$-round public-coin interactive proof Π for relation $\mathfrak{R}$, where $\mathcal{V}$ samples the i-th challenge from a set of cardinality $N_i \geq k_i$ for $1 \leq i \leq \mu$, is $\mathbf{k}$-out-of-$\mathbf{N}$ special-sound if there exists a polynomial time algorithm that, on input a statement x and a $\mathbf{k}$-tree of accepting transcripts, outputs a witness $w \in \mathfrak{R}(x)$. We also say Π is $\mathbf{k}$-special-sound.

In contrast to the extractor $\mathcal{E}$ of Definition 2.27 that has only oracle access to the prover, the special-soundness algorithm obtains the transcripts directly. For this reason, it is nontrivial to show that special-soundness implies knowledge

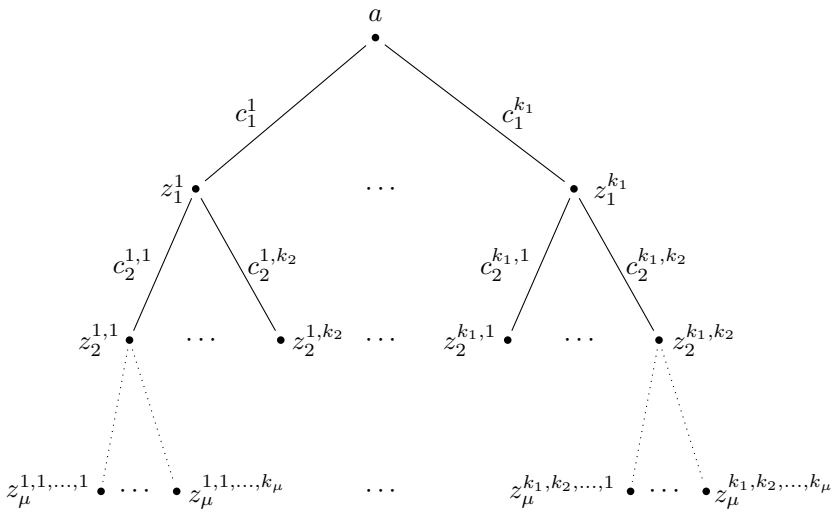

Figure 2.1: $(k_1, \ldots, k_\mu)$-tree of transcripts [ACK21].

soundness. While it is well known that for 3-round protocols special-soundness implies knowledge soundness, previously there was no known generalization to $2\mu+1$-round protocols. In Chapter 6 we show that, also for multi-round protocols, special-soundness tightly implies knowledge soundness.

2.8.3 Zero-Knowledge

In many applications, the prover $\mathcal{P}$ wishes to convince the verifier $\mathcal{V}$ without releasing any information besides the veracity of the claim. In particular, a protocol execution should not reveal any additional information about the secret witness $w \in \mathfrak{R}(x)$, even if the verifier behaves maliciously. An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ that satisfies this security property is said to be *zero-knowledge* and also called a *zero-knowledge proof* (ZKP).

In Definition 2.35 this security property is formalized by means of a so-called *simulator*. A simulator takes as input the public statement x and outputs protocol transcripts that are distributed statistically close to transcripts generated by interactions with the honest prover $\mathcal{P}$. The existence of a simulator shows that a (potentially dishonest) verifier $\mathcal{V}^*$ can generate transcripts *without* interacting with the honest prover $\mathcal{P}$, i.e., the interactions with $\mathcal{P}$ do not reveal any information that $\mathcal{V}^*$ could not have obtained on its own.

Definition 2.35 (Zero-Knowledge)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is said to be (statistical) *zero-knowledge* (ZK) if, for every (potentially dishonest) polynomial time verifier $\mathcal{V}^*$, there exists a polynomial time simulator $\mathcal{S}^*$ such that the following families of random variables are statistically indistinguishable:

- $\{\mathsf{view}^{\mathcal{P}}_{\mathcal{V}^*}(x; w) : (x; w) \in \mathfrak{R}\}$, where $\mathsf{view}^{\mathcal{P}}_{\mathcal{V}^*}(x; w)$ describes $\mathcal{P}$'s messages and

$\mathcal{V}^*$'s random tape when evaluating $(\mathcal{P}, \mathcal{V}^*)$ on input $(x; w)$;

- $\{\mathcal{S}^*(x) : (x; w) \in \mathfrak{R}\}$.

If these families of random variables are only computationally indistinguishable, Π is said to be *computationally* zero-knowledge.

Remark 2.7. Sometimes it is convenient to make the statistical distance between the distributions of Definition 2.35 explicit. In this case, we say Π is δ-statistical zero-knowledge, for some $\delta \colon \mathbb{N} \to [0, 1]$, if

$$\Delta\big(\mathsf{view}^{\mathcal{P}}_{\mathcal{V}^*}(x; w), \mathcal{S}^*(x)\big) \leq \delta(|x|) \quad \forall (x; w) \in \mathfrak{R}.$$

We also consider a weaker notion of zero-knowledge: *honest-verifier zero-knowledge*. This notion only requires the existence of a simulator for the *honest* verifier $\mathcal{V}$, i.e., a simulator that outputs transcripts distributed statistically close to transcripts of *honest* executions of Π. Typically, a prover cannot distinguish between interactions with honest and dishonest verifiers, therefore in most applications this weaker security property does not suffice. However, there exist generic transformations that transform certain classes of HVZK interactive proofs, such as public-coin ones, into zero-knowledge interactive proofs [OVY93; Dam93; DGO+95]. Alternatively, public-coin interactive proofs can be made non-interactive by applying the Fiat-Shamir transform [FS86]. In this transformation, the verifier's messages (challenges) are replaced by random oracle queries. In the Fiat-Shamir mode, honest-verifier zero-knowledge does suffice. For these reasons, it is often enough to show that an interactive proof is honest-verifier zero-knowledge.

Definition 2.36 ((Special) Honest-Verifier Zero-Knowledge)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is said to be (statistical) *honest-verifier zero-knowledge* (HVZK) if there exists a polynomial time simulator $\mathcal{S}$ such that the following families of random variables are statistically indistinguishable:

- $\{\mathsf{view}^{\mathcal{P}}_{\mathcal{V}}(x; w) : (x; w) \in \mathfrak{R}\}$, where $\mathsf{view}^{\mathcal{P}}_{\mathcal{V}}(x; w)$ describes $\mathcal{P}$'s messages and $\mathcal{V}$'s random tape when evaluating $\Pi = (\mathcal{P}, \mathcal{V})$ on input $(x; w)$;
- $\{\mathcal{S}(x) : (x; w) \in \mathfrak{R}\}$.

If $\Delta\big(\mathsf{view}^{\mathcal{P}}_{\mathcal{V}}(x; w), \mathcal{S}(x)\big) = 0$ for all $(x; w) \in \mathfrak{R}$, Π is said to be *perfectly* HVZK. If these families of random variables are only computationally indistinguishable, Π is said to be *computationally* HVZK. Further, if the simulator proceeds by first sampling the verifier's messages uniformly at random, Π is said to be *special* honest-verifier zero-knowledge (SHVZK).

Finally, we consider yet another relaxation of the zero-knowledge property. For some interactive proofs $\Pi = (\mathcal{P}, \mathcal{V})$, honest executions do reveal information about the secret witness $w \in \mathfrak{R}(x)$, but *only* if the prover $\mathcal{P}$ aborts during the protocol execution. These protocols admit a simulator that can simulate *non-aborting* transcripts and are said to be *non-abort honest-verifier zero-knowledge* (NA-HVZK). It is typically straightforward to transform an NA-HVZK interactive proof into

one that is HVZK. Moreover, in the non-interactive Fiat-Shamir instantiation of a public-coin interactive proof, aborting executions are never published, and the weaker notion of NA-HVZK suffices. In the literature NA-HVZK is often simply referred to as HVZK. We use a different notation to emphasize the difference.

Definition 2.37 (Non-Abort Honest-Verifier Zero-Knowledge)**.** An interactive proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is said to be (statistical) *non-abort honest-verifier zero-knowledge* (NA-HVZK) if there exists a polynomial time simulator $\mathcal{S}$ such that the following families of random variables are statistically indistinguishable:

- $\{\mathsf{NA\text{-}view}_{\mathcal{V}}^{\mathcal{P}}(x;w) : (x;w) \in \mathfrak{R}\}$, where $\mathsf{NA\text{-}view}_{\mathcal{V}}^{\mathcal{P}}(x;w)$ describes $\mathcal{P}$'s messages and $\mathcal{V}$'s random tape when evaluating $\Pi = (\mathcal{P}, \mathcal{V})$ on input $(x;w)$, conditioned on $\mathcal{P}$ not aborting;
- $\{\mathcal{S}(x) : (x;w) \in \mathfrak{R}\}$.

If the simulator proceeds by first sampling the verifier's messages uniformly at random, then Π is said to be non-abort *special* honest-verifier zero-knowledge (NA-SHVZK).

Remark 2.8. Definition 2.37 allows the abort probability of an honest prover to depend on the secret witness $w \in \mathfrak{R}(x)$. However, the generic transformations from NA-HVZK to HVZK typically require the abort probability to be essentially independent of the witness w. Moreover, also in the non-interactive Fiat-Shamir mode, it is preferable to have an abort probability independent of the witness; otherwise, the non-interactive proof might be susceptible to side-channel attacks, e.g., timing attacks. For this reason, in addition, we typically require that the abort probability of an honest prover is essentially independent of the witness w.

2.9 Non-Interactive Proofs in the Random Oracle Model

In the *random oracle model* (ROM), algorithms have oracle access to a function $\mathsf{RO}\colon \{0,1\}^* \to \{0,1\}^\eta$, called a random oracle, sampled uniformly at random from the set of functions with domain $\{0,1\}^*$ and codomain $\{0,1\}^\eta$ for some $\eta \in \mathbb{N}$. A random oracle RO is implicitly instantiated by *lazy sampling*, i.e., every time the random oracle is queried on a new input $x \in \{0,1\}^*$, the evaluation $\mathsf{RO}(x) \in \{0,1\}^\eta$ is sampled uniformly at random and fixed from that point onward. In particular, if the random oracle is queried on the same input x as before, possibly by a different algorithm, it will return the same output $\mathsf{RO}(x)$.

A random oracle $\mathsf{RO}\colon \{0,1\}^* \to \{0,1\}^\eta$ outputs bitstrings of length η. However, the codomain of a random oracle is adapted easily. For instance, if one requires bitstrings of length $\eta' \leq \eta$, the evaluation $\mathsf{RO}(x)$ can be truncated to its first η' bits and, if one requires bitstrings of length $k \cdot \eta$, simply define

$$\mathsf{RO}'\colon \{0,1\}^* \to \{0,1\}^{k\cdot\eta}, \quad x \mapsto \mathsf{RO}(1\|x)\|\cdots\|\mathsf{RO}(k\|x)\,,$$

where $i\|x$ denotes the bitstring x prepended with the bit decomposition of $i \in \mathbb{N}$. In fact, the random oracle $\mathsf{RO}\colon \{0,1\}^* \to \{0,1\}^\eta$ can be adapted to output elements in any finite set $\mathcal{Y}$. Therefore, we allow the codomain of a random oracle

to be an arbitrary finite set $\mathcal{Y}$. Moreover, for convenience, we sometimes leave the codomain $\mathcal{Y}$ implicit and write $\mathcal{RO}$ for the set of all random oracles. Further, to avoid technical difficulties, we sometimes limit the domain from $\{0,1\}^*$ to $\{0,1\}^{\leq u}$, the finite set of all bitstrings of length at most u, for a sufficiently large $u \in \mathbb{N}$.

An algorithm $\mathcal{A}$ with oracle access to a random oracle RO, which is denoted as $\mathcal{A}^{\mathsf{RO}}$, is called a *random oracle algorithm*. The algorithm $\mathcal{A}$ is said to be a *Q-query* random oracle algorithm if, for all inputs x, random tapes r and random oracles RO, $\mathcal{A}^{\mathsf{RO}}$ makes at most Q queries to RO.

The Fiat-Shamir transformation (Section 2.9.2) allows *public-coin* interactive proofs $\Pi = (\mathcal{P}, \mathcal{V})$ to be made non-interactive in the random oracle model. The high level idea is that the verifier's challenges are replaced by random oracle queries. This way the prover can generate a proof for knowledge of a witness $w \in \mathfrak{R}(x)$ without interacting with the verifier. The resulting protocol is called a *non-interactive random oracle proof* (NIROP). Vice versa, a non-interactive random oracle proof also corresponds to an interactive proof, obtained by replacing the random oracle queries with challenges sampled by the verifier.

Definition 2.38 (Non-Interactive Random Oracle Proof)**.** A *non-interactive random oracle proof* (NIROP) for relation $\mathfrak{R}$ is a pair $\Pi = (\mathcal{P}, \mathcal{V})$ of (probabilistic) random-oracle algorithms, a prover $\mathcal{P}$ and a polynomial time verifier $\mathcal{V}$, such that: Given $(x; w) \in \mathfrak{R}$ and access to a random oracle RO, the prover $\mathcal{P}^{\mathsf{RO}}(x; w)$ outputs a proof π. Given $x \in \{0,1\}^*$, a purported proof π, and access to a (random) oracle RO, the verifier $\mathcal{V}^{\mathsf{RO}}(x, \pi)$ outputs 0 to reject or 1 to accept the proof.

Remark 2.9. Standard techniques for "domain separation" allow multiple random oracles $\mathsf{RO}_1, \ldots, \mathsf{RO}_k$ to be constructed from a single one [BR93], e.g., by defining $\mathsf{RO}_i(x) := \mathsf{RO}(i \| x)$ for all $1 \leq i \leq k$. For this reason, if required or convenient, we allow the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ of a NIROP $\Pi = (\mathcal{P}, \mathcal{V})$ to have access to multiple independent random oracles $\mathsf{RO}_1, \ldots, \mathsf{RO}_k$, possibly with different codomains.

The following definition is a natural adaptation of the completeness property for interactive proofs. Note that here, besides $\mathcal{P}$'s and $\mathcal{V}$'s randomness, the probability is over the randomness of the random oracle RO.

Definition 2.39 (Completeness - NIROP)**.** A non-interactive random oracle proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *complete* with completeness error $\rho \colon \mathbb{N} \to [0,1]$ if, for all $(x; w) \in \mathfrak{R}$,

$$\Pr\big((\mathcal{P}^{\mathsf{RO}}(w), \mathcal{V}^{\mathsf{RO}})(x) = \mathsf{reject} : \mathsf{RO} \leftarrow_R \mathcal{RO}\big) \leq \rho(|x|)\,.$$

If $\rho(|x|) = 0$ for all x, $(\mathcal{P}, \mathcal{V})$ is said to be perfectly complete.

Similarly, the soundness property of interactive proofs can be adapted to a soundness property for non-interactive random oracle proofs. Note that the soundness error $\sigma(|x|, Q)$ is allowed to depend on the query complexity Q of the prover $\mathcal{P}^*$ attacking the considered NIROP. For many NIROPs, it is indeed the case that the success probability of a cheating prover $\mathcal{P}^*$ increases with the number of random oracle queries Q admitted to the prover $\mathcal{P}^*$.

Definition 2.40 (Soundness - NIROP)**.** A non-interactive random oracle proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *sound* with soundness error $\sigma\colon \mathbb{N} \times \mathbb{N} \to [0,1]$ if for all $x \notin L_{\mathfrak{R}}$ and every Q-query prover $\mathcal{P}^*$,

$$\Pr\big((\mathcal{P}^{*,\mathsf{RO}}, \mathcal{V}^{\mathsf{RO}})(x) = \mathsf{accept} : \mathsf{RO} \leftarrow_R \mathcal{RO}\big) \leq \sigma(|x|, Q)\,.$$

If this property only holds for (probabilistic) polynomial time (i.e., computationally bounded) provers $\mathcal{P}^*$, then Π is said to be *computationally* sound.

Also the knowledge soundness definition can be adapted to non-interactive random oracle proofs. As before, knowledge soundness requires the existence of an extractor $\mathcal{E}$ that, given input x and oracle access to a prover $\mathcal{P}^*$, aims to output a witness $w \in \mathfrak{R}(x)$. However, a crucial difference with Definition 2.27, for interactive proofs, is that now the prover $\mathcal{P}^*$ attacking the considered NIROP is a *random oracle* algorithm, instead of a "normal" algorithm. Giving the knowledge extractor $\mathcal{E}$ oracle access to the random oracle algorithm $\mathcal{P}^*$ means that $\mathcal{E}$ can invoke $\mathcal{P}^{*,\mathsf{RO}}$ for any random oracle RO. More precisely, $\mathcal{E}$ observes all the random oracle queries made by $\mathcal{P}^*$ and is free to decide how to answer these queries. We also say that $\mathcal{E}$ implements RO for $\mathcal{P}^*$. Hence, instead of extracting a witness by controlling the verifier's challenge, an extractor for NIROPs aims to output a witness by controlling the random oracle responses.

Definition 2.41 (Knowledge Soundness - NIROP)**.** A non-interactive random oracle proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *knowledge sound* with *knowledge error* $\kappa\colon \mathbb{N} \times \mathbb{N} \to [0,1]$, if there exists a positive polynomial q and an algorithm $\mathcal{E}$, called a *knowledge extractor*, with the following properties: The extractor $\mathcal{E}^{\mathcal{P}^*}(x)$, given input x and oracle access to a (potentially dishonest) Q-query random oracle prover $\mathcal{P}^*$, runs in an expected number of steps that is polynomial in $|x|$ and Q and outputs a witness $w \in \mathfrak{R}(x)$ with probability

$$\Pr\big((x; \mathcal{E}^{\mathcal{P}^*}(x)) \in \mathfrak{R}\big) \geq \frac{\epsilon(x, \mathcal{P}^*) - \kappa(|x|, Q)}{q(|x|)}\,,$$

where $\epsilon(x, \mathcal{P}^*) = \Pr\big(\mathcal{V}^{\mathsf{RO}}(x, \mathcal{P}^{*,\mathsf{RO}}(x)) = \mathsf{accept} : \mathsf{RO} \leftarrow_R \mathcal{RO}\big)$.

It is easy to see that any cheating strategy for the interactive proof corresponding to a NIROP gives a cheating strategy for the NIROP itself that succeeds with exactly the same probability. Hence, $\kappa_{\mathrm{IP}}(|x|) \leq \kappa_{\mathrm{NI}}(|x|, Q)$ for all $x \in \{0,1\}^*$ and $Q \in \mathbb{N}$, where $\kappa_{\mathrm{IP}}(|x|)$ and $\kappa_{\mathrm{NI}}(|x|, Q)$ are the knowledge errors of the interactive and non-interactive proofs, respectively. For this reason we also refer to the ratio

$$\frac{\kappa_{\mathrm{NI}}(|x|, Q)}{\kappa_{\mathrm{IP}}(|x|)}$$

as the security loss of the NIROP. We are typically interested in how this security loss scales as a function of Q.

Finally, let us consider the zero-knowledge property. As for interactive proofs, a non-interactive random oracle proof $\Pi = (\mathcal{P}, \mathcal{V})$ is said to be zero-knowledge if there exists a simulator that aims to output a proof π that is indistinguishable

from honestly generated proofs. To this end, it is given as input a statement x and oracle access to a random oracle RO. However, in contrast to honest provers, the simulator is allowed to *reprogram* the random oracle $\mathsf{RO}\colon \{0,1\}^* \to \mathcal{Y}$ at arbitrary inputs. Let $L = \{(x_1, y_1), \ldots, (x_k, y_k)\} \subseteq \{0,1\}^* \times \mathcal{Y}$ with pairwise distinct x_i, then we write $\mathsf{RO}[L]$ for the random oracle that is reprogrammed in L, i.e.,

$$\mathsf{RO}[L](x) = \begin{cases} y_i, & \text{if } \exists i \text{ s.t. } x_i = x, \\ \mathsf{RO}(x), & \text{otherwise.} \end{cases}$$

This zero-knowledge property for non-interactive random oracle proofs is formalized in the following definition. It is easily seen that replacing the challenges of an honest-verifier zero-knowledge interactive proof by random oracle queries results in a NIROP that is zero-knowledge.

Definition 2.42 (Zero-Knowledge - NIROP)**.** A non-interactive random oracle proof $\Pi = (\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is said to be (statistical) *zero-knowledge* if there exists a polynomial time random oracle simulator $\mathcal{S}$ such that, for every distinguisher $\mathcal{D}\colon \{0,1\}^* \to \{0,1\}$, the two families $\{X(x;w) : (x;w) \in \mathfrak{R}\}$ and $\{Y(x;w) : (x;w) \in \mathfrak{R}\}$ of distributions defined as

- $X(x;w) = \mathcal{D}^{\mathsf{RO}[L]}(\pi)$, where $(\pi, L) \leftarrow \mathcal{S}^{\mathsf{RO}}(x)$ and $\mathsf{RO} \leftarrow_R \mathcal{RO}$;
- $Y(x;w) = \mathcal{D}^{\mathsf{RO}}(\pi)$, where $\pi \leftarrow \mathcal{P}^{\mathsf{RO}}(x;w)$ and $\mathsf{RO}_R \leftarrow \mathcal{RO}$;

are statistically indistinguishable. If the above only holds for polynomial time distinguishers $\mathcal{D}$, Π is said to be *computationally* zero-knowledge.

2.9.1 Adaptive Knowledge Soundness

Thus far, knowledge soundness has been defined with respect to *static* or *non-adaptive* provers $\mathcal{P}^*$ attacking the considered (non-)interactive proof for a *fixed* statement x. However, in many practical scenarios the dishonest provers are free to *choose* the statement x adaptively. Hence, in these cases static security is not sufficient. For *interactive* proofs, it is well known that static knowledge soundness implies adaptive knowledge soundness. However, this does not carry over to non-interactive proofs. For instance, it is easy to see that the static Fiat-Shamir transformation (see Definition 2.44) is in general not adaptively sound.

For this reason, let us formalize adaptive knowledge soundness for non-interactive random oracle proofs. An adaptive prover $\mathcal{P}^{\mathsf{a}}$ attacking the considered NIROP is given oracle access to a random oracle RO and outputs a statement x of fixed length $|x| = n$ together with a proof π. As in the static definition, adaptive knowledge soundness requires the existence of a knowledge extractor. However, formalizing the requirements of this extractor introduces some subtle issues. Namely, because $\mathcal{P}^{\mathsf{a}}$ chooses the statement x adaptively, it is not immediately clear for which statement the extractor should extract a witness. For instance, granting the extractor the same freedom of adaptively choosing the statement x, for which it needs to extract a witness w, renders knowledge extraction trivial; the extractor could simply output an arbitrary statement-witness pair $(x;w)$. For this reason, we require the extractor to output statement-witness pairs $(x;w)$ corresponding

to the *valid* pairs (x, π) output by the adaptive prover $\mathcal{P}^{\mathsf{a}}$. To formalize these requirements, we also write (x, π, v), with $v \in \{0,1\}$ indicating whether π is a valid proof for statement x. Given this notation, the extractor should output a triple (x, π, v) with the same distribution as the triples (x, π, v) produced by $\mathcal{P}^{\mathsf{a}}$; furthermore, if π is a valid proof for statement x, i.e., $v = 1$, then the extractor should additionally aim to output a witness $w \in \mathfrak{R}(x)$. As before, the success probability of the extractor is allowed to depend on the success probability of $\mathcal{P}^{\mathsf{a}}$. Finally, to ensure that the knowledge extractor can be used in compositional settings, where the NIROP is deployed as a component of a larger protocol, the prover $\mathcal{P}^{\mathsf{a}}$ is also allowed to additionally output arbitrary auxiliary information $\mathsf{aux} \in \{0,1\}^*$, and the extractor is then required to simulate the tuple $(x, \pi, \mathsf{aux}, v)$, rather than the triple (x, π, v). The following definition formalizes adaptive knowledge soundness along these lines. For alternative definitions see, e.g., [Unr17; DFM+19].

Definition 2.43 (Adaptive Knowledge Soundness - NIROP)**.** A non-interactive random oracle proof $(\mathcal{P}, \mathcal{V})$ for relation $\mathfrak{R}$ is *adaptively knowledge sound* with *knowledge error* $\kappa\colon \mathbb{N} \times \mathbb{N} \to [0,1]$, if there exists a positive polynomial q and an algorithm $\mathcal{E}$, called a *knowledge extractor*, with the following properties: The extractor, given input $n \in \mathbb{N}$ and oracle access to any adaptive Q-query random oracle prover $\mathcal{P}^{\mathsf{a}}$ that outputs statements x with $|x| = n$, runs in an expected number of steps that is polynomial in n and Q and outputs a tuple $(x, \pi, \mathsf{aux}, v; w)$ such that $\{(x, \pi, \mathsf{aux}, v) : (x, \pi, \mathsf{aux}) \leftarrow \mathcal{P}^{\mathsf{a},\mathsf{RO}} \wedge v \leftarrow \mathcal{V}^{\mathsf{RO}}(x, \pi)\}$ and $\{(x, \pi, \mathsf{aux}, v) : (x, \pi, \mathsf{aux}, v; w) \leftarrow \mathcal{E}^{\mathcal{P}^{\mathsf{a}}}(n)\}$ are identically distributed and

$$\Pr\big(v = \mathsf{accept} \wedge (x; w) \in \mathfrak{R} : (x, \pi, \mathsf{aux}, v; w) \leftarrow \mathcal{E}^{\mathcal{P}^{\mathsf{a}}}(n)\big) \geq \frac{\epsilon(\mathcal{P}^{\mathsf{a}}) - \kappa(n, Q)}{q(n)}\,,$$

where $\epsilon(\mathcal{P}^{\mathsf{a}}) = \Pr\big(\mathcal{V}^{\mathsf{RO}}(x, \pi) = 1 : (x, \pi) \leftarrow \mathcal{P}^{\mathsf{a},\mathsf{RO}}\big)$. Here, $\mathcal{E}$ implements RO for $\mathcal{P}^{\mathsf{a}}$; in particular, $\mathcal{E}$ can arbitrarily program RO. Moreover, the randomness is over the randomness of $\mathcal{E}$, $\mathcal{V}$, $\mathcal{P}^{\mathsf{a}}$ and RO.

Remark 2.10. We note that, while the tuple $(x, \pi, \mathsf{aux}, v)$ is required to have the same distribution for $\mathcal{P}^{\mathsf{a}}$ and $\mathcal{E}(n)$, by default the respective executions of $\mathcal{P}^{\mathsf{a}}$ and $\mathcal{E}(n)$ give rise to two different probability spaces. Looking ahead though, we remark that the extractor that we eventually construct (Section 6.6) first does an honest run of $\mathcal{P}^{\mathsf{a}}$ by faithfully simulating the answers to $\mathcal{P}^{\mathsf{a}}$'s random oracle queries (this produces the tuple $(x, \pi, \mathsf{aux}, v)$ that $\mathcal{E}(n)$ eventually outputs and which so trivially has the right distribution), and then, if π is a valid proof, $\mathcal{E}(n)$ starts rewinding $\mathcal{P}^{\mathsf{a}}$ and reprogramming the random oracle to try to find enough valid proofs to compute a witness. Thus, in this sense, we can then say that $\mathcal{E}(n)$ aims to find a witness $w \in \mathfrak{R}(x)$ for *the statement x output by $\mathcal{P}^{\mathsf{a}}$*.

2.9.2 Fiat-Shamir Transformation

The Fiat-Shamir transformation [FS86] turns a public-coin interactive proof into a non-interactive random oracle proof (NIROP). The general idea is to compute the i-th challenge c_i as a hash (i.e., the output of a random oracle which in practice is a hash function) of the i-th prover message a_i and (some part of) the previous communication transcript. For a Σ-protocol, the challenge c is computed as $c =$

$H(a)$, or as $c = H(x, a)$, where the former is sufficient for *static* security, where the statement x is given as input to the dishonest prover, and the latter is necessary for *adaptive* security, where the dishonest prover can choose the statement x for which it wants to forge a proof.

For multi-round public-coin interactive proofs, there is some degree of freedom in the computation of the i-th challenge. For concreteness and simplicity, we consider a particular version where all previous prover messages are hashed along with the current message. As for Σ-protocols, we consider a static and an adaptive variant of this version of the Fiat-Shamir transformation. In contrast to the static variant, the adaptive Fiat-Shamir transformation includes the statement x in all hash function evaluations. If it is not made explicit which variant is used, the considered result holds for both variants.

Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $(2\mu + 1)$-round public-coin interactive proof, where the challenge from the i-th round is sampled from set $\mathcal{C}_i$. For simplicity, we consider μ random oracles $\mathsf{RO}_i \colon \{0,1\}^{\leq u} \to \mathcal{C}_i$ that map into the respective challenge spaces.

Definition 2.44 (Fiat-Shamir Transformation)**.** Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a public-coin interactive proof. The static Fiat-Shamir transformation $\mathsf{FS}[\Pi] = (\mathcal{P}_{\mathsf{fs}}, \mathcal{V}_{\mathsf{fs}})$ is the NIROP where $\mathcal{P}_{\mathsf{fs}}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_\mu}(x; w)$ runs $\mathcal{P}(x; w)$, but instead of asking the verifier for the challenge c_i on message a_i, the challenges are computed as

$$c_i = \mathsf{RO}_i(a_1, \ldots, a_{i-1}, a_i)\,; \tag{2.3}$$

the output is then the proof $\pi = (a_1, \ldots, a_{\mu+1})$. On input a statement x and a proof $\pi = (a_1, \ldots, a_{\mu+1})$, $\mathcal{V}_{\mathsf{fs}}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_\mu}(x, \pi)$ accepts if, for c_i as above $\mathcal{V}$ accepts the transcript $(a_1, c_1, \ldots, a_\mu, c_\mu, a_{\mu+1})$ on input x.

If the challenges are computed as

$$c_i = \mathsf{RO}_i(x, a_1, \ldots, a_{i-1}, a_i)\,; \tag{2.4}$$

the resulting NIROP is referred to as the *adaptive* Fiat-Shamir transformation.

By means of reducing the security of other variants of the Fiat-Shamir transformation to Definition 2.44, appropriately adjusted versions of our results also apply to other variants of doing the "chaining" (Equations 2.3 and 2.4) in the Fiat-Shamir transformation, for instance when c_i is computed as $c_i = \mathsf{RO}_i(i, c_{i-1}, a_i)$, or $c_i = \mathsf{RO}_i(x, i, c_{i-1}, a_i)$, where c_0 is the empty string.

2.10 Secret-Sharing Schemes

A secret-sharing scheme allows a secret to be distributed amongst a set of players, such that sufficiently small subsets of players do not have any information about the secret, while large enough subsets are able to reconstruct the secret. A secret-sharing scheme is said to be *linear* if its secret space is a finite field $\mathbb{F}$ and every share can be computed as the linear combination of the secret $s \in \mathbb{F}$ and a number of random field elements. Because a more general treatment is not required in this dissertation, we will restrict ourselves to linear secret-sharing schemes (LSSSs) for which each share is a single field element. For a more general definition, in terms

of error correcting codes and allowing shares to consist of multiple field elements, we refer to [CDN15]. Further, a *packed* LSSS, also called a ramp scheme, considers secret vectors $\mathbf{x} \in \mathbb{F}^m$, i.e. this notion generalizes the secret space dimension from $m = 1$ to arbitrary $m \in \mathbb{N}$.

Definition 2.45 (Packed Linear Secret-Sharing Scheme)**.** Let $m, n, t \in \mathbb{N}$ and $\mathbb{F}$ a finite field. A linear secret-sharing scheme $\mathcal{S}$ for sharing m-dimensional vectors $\mathbf{x} \in \mathbb{F}^m$ amongst a set of n players is defined by a matrix $M \in \mathbb{F}^{n\times(m+t)}$. A secret sharing of $\mathbf{x} \in \mathbb{F}^m$ is computed by sampling a vector $\mathbf{r} \leftarrow_R \mathbb{F}^t$ uniformly at random and outputting the share vector

$$[\mathbf{x}; \mathbf{r}]_{\mathcal{S}} = M \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} \in \mathbb{F}^n .$$

If the scheme $\mathcal{S}$ is clear from context, we simply write $[\mathbf{x}; \mathbf{r}]$.

Every player in a linear secret-sharing scheme $\mathcal{S}$ thus corresponds to one row of the matrix M. For all k-subsets[5] $A \subseteq \{1, \ldots, n\}$ of players, $M_A \in \mathbb{F}^{k\times(m+t)}$ is defined to be the matrix consisting of the rows of the players in A. Hence,

$$M_A \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} \in \mathbb{F}^k$$

is the vector containing the shares of the players in A. The privacy property of a secret-sharing scheme states that sufficiently small subsets A are not able to deduce any information about the secret vector $\mathbf{x} \in \mathbb{F}^m$ from their shares. These subsets are also referred to as *unqualified*, and the set of all unqualified subsets is referred to as the *adversary structure*.

Definition 2.46 (Secret Sharing - Privacy)**.** Let $m, n, t, p \in \mathbb{N}$ with $p \leq n$ and $\mathbb{F}$ a finite field. A linear secret-sharing scheme $\mathcal{S}$, defined by the matrix $M \in \mathbb{F}^{n\times(m+t)}$, is said to have *p-privacy* if for every p-subset $A \subseteq \{1, \ldots, n\}$, the distribution

$$\left\{ M_A \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} \in \mathbb{F}^p : \mathbf{r} \leftarrow_R \mathbb{F}^t \right\}$$

is independent of $\mathbf{x} \in \mathbb{F}^m$.

The reconstruction property of a secret-sharing scheme states that sufficiently large subsets of players are able to reconstruct the secret given their shares. These subsets are also referred to as *qualified*, and the set of all qualified subsets is referred to as the *access structure*. In this dissertation, the definitions are restricted to *threshold* access structures, i.e., an access structure containing all subsets of a certain minimal cardinality. For a treatment of more general access structures we refer to [CDN15].

Definition 2.47 (Secret Sharing - Reconstruction)**.** Let $m, n, t, r \in \mathbb{N}$ with $r \leq n$ and $\mathbb{F}$ a finite field. A linear secret-sharing scheme $\mathcal{S}$, defined by the

[5] A k-subset is a subset of cardinality k.

matrix $M \in \mathbb{F}^{n\times(m+t)}$, is said to have *r-reconstruction* if, for every r-subset $A \subseteq \{1,\dots,n\}$, $\mathbf{x} \in \mathbb{F}^m$ is uniquely determined by

$$M_A \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} \in \mathbb{F}^r .$$

It can be shown that an LSSS with r-reconstruction, for every r-subset $A \subseteq \{1,\dots,n\}$ admits a matrix $U_A \in \mathbb{F}^{m\times r}$ such that

$$U_A M_A \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} = \mathbf{x} \in \mathbb{F}^m ,$$

for all $\mathbf{x}$ and $\mathbf{r}$ [CDN15]. Hence, also the reconstruction of a secret from its shares is a linear operation.

If the secret space dimension m of $\mathcal{S}$ equals 1, every subset $A \subseteq \{1,\dots,n\}$ of players is either qualified or unqualified [CDN15, Theorem 6.8]. In this case, there exists a $k \in \mathbb{N}$ such that $\mathcal{S}$ has k-reconstruction and $(k-1)$-privacy, and $\mathcal{S}$ is called a k-out-of-n or a (k,n)-secret-sharing scheme. Note that the above does not hold for arbitrary $m \in \mathbb{N}$. More precisely, if $m > 1$, there might exist subsets A that are neither qualified nor unqualified.

The component-wise product

$$[\mathbf{x};\mathbf{r_x}]_{\mathcal{S}} * [\mathbf{y};\mathbf{r_y}]_{\mathcal{S}} \in \mathbb{F}^n$$

of two share-vectors turns out to be a linear secret sharing of the component-wise product $\mathbf{x} * \mathbf{y} \in \mathbb{F}^m$ of the two secret vectors, however, with respect to a different LSSS $\widehat{\mathcal{S}}$. Namely, let $\widehat{M} \in \mathbb{F}^{n\times(m+t)^2}$ be such that its i-th row is the tensor product of the i-th row of M with itself. Then it is easily seen that

$$[\mathbf{x};\mathbf{r_x}]_{\mathcal{S}} * [\mathbf{y};\mathbf{r_y}]_{\mathcal{S}} = M \begin{pmatrix} \mathbf{x} \\ \mathbf{r_x} \end{pmatrix} * M \begin{pmatrix} \mathbf{y} \\ \mathbf{r_y} \end{pmatrix} = \widehat{M} \left(\begin{pmatrix} \mathbf{x} \\ \mathbf{r_x} \end{pmatrix} \otimes \begin{pmatrix} \mathbf{y} \\ \mathbf{r_y} \end{pmatrix} \right) .$$

Since the vector $\mathbf{x} \otimes \mathbf{y}$ contains the component-wise product $\mathbf{x} * \mathbf{y}$ as a subvector, the above equation shows that $[\mathbf{x};\mathbf{r_x}]_{\mathcal{S}} * [\mathbf{y};\mathbf{r_y}]_{\mathcal{S}}$ is indeed a secret sharing of $\mathbf{x} * \mathbf{y}$ with respect to the LSSS $\widehat{\mathcal{S}}$ defined by $\widehat{M}$. The scheme $\mathcal{S}$ is said to have *product-reconstruction* if $\widehat{S}$ has the reconstruction property. In this case, $\mathcal{S}$ is also said to be *multiplicative*.

Definition 2.48 (Secret Sharing - Product-Reconstruction)**.** Let $m, n, t, R \in \mathbb{N}$ with $R \leq n$ and $\mathbb{F}$ a finite field. A linear secret-sharing scheme $\mathcal{S}$, defined by the matrix $M \in \mathbb{F}^{n\times(m+t)}$, is said to have *R-product-reconstruction* if the secret-sharing scheme $\widehat{\mathcal{S}}$, defined as above by the matrix $\widehat{M}$, has R-reconstruction.

2.10.1 Shamir Secret-Sharing

Shamir's scheme [Sha79] is perhaps the best-known example of a linear secret sharing scheme. Its secret space is a finite field $\mathbb{F}$ with at least $n+1$ elements[6],

[6] When additionally taking the point at infinity into account, this requirement can be relaxed to $|\mathbb{F}| \geq n$. For more details see [CDN15].

where n is the number of players. Instantiated with privacy parameter $1 \leq p \leq n$, it is defined by the Vandermonde matrix

$$M = \begin{pmatrix} 1 & \alpha_1 & \cdots & \alpha_1^p \\ 1 & \alpha_2 & \cdots & \alpha_2^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \cdots & \alpha_n^p \end{pmatrix} \in \mathbb{F}^{n \times p+1} ,$$

where $\alpha_1, \ldots, \alpha_n \in \mathbb{F} \setminus \{0\}$ are pairwise distinct. A Shamir secret sharing $[s; \mathbf{r}]$ of $s \in \mathbb{F}$ thus corresponds to n evaluations of the polynomial $f(X) = s + r_1 X + \cdots r_p X^p \in \mathbb{F}[X]$ of degree at most p, i.e.,

$$[s; \mathbf{r}] = M \begin{pmatrix} s \\ \mathbf{r} \end{pmatrix} = (f(\alpha_1), \ldots, f(\alpha_n)) \in \mathbb{F}^n .$$

By Lagrange interpolation it follows that the polynomial $f(X)$ is uniquely determined by any set containing at least $p + 1$ of its evaluations. Hence, this instantiation of Shamir's secret-sharing scheme has $(p + 1)$-reconstruction. Moreover, again by Lagrange interpolation, for any $s \in \mathbb{F}$ and any p-subset $A \subseteq \{1, \ldots, n\}$, the mapping

$$L \colon \mathbb{F}^p \to \mathbb{F}^p, \quad \mathbf{r} \mapsto M_A \begin{pmatrix} s \\ \mathbf{r} \end{pmatrix}$$

is bijective. Therefore, it follows that this scheme has p-privacy, i.e., it is a $(p+1)$-out-of-n secret sharing scheme.

Further, observe that the component-wise product of two share-vectors equals

$$[s_1; \mathbf{r}_1] * [s_2; \mathbf{r}_2] = (f(\alpha_1), \ldots, f(\alpha_n)) * (g(\alpha_1), \ldots, g(\alpha_n)) = (h(\alpha_1), \ldots, h(\alpha_n)) ,$$

for polynomials $f(X)$, $g(X)$ and $h(X) = f(X)g(X)$. Hence, since $h(X)$ is of degree at most $2p$, this secret-sharing scheme has $(2p + 1)$-product-reconstruction, i.e., if $2p + 1 \leq n$, it is multiplicative.

In the above, the secret $s \in \mathbb{F}$ is allocated to the constant coefficient of the secret-sharing polynomial $f(X)$, i.e., $s = f(0)$. Equivalently, the secret can be allocated to any other evaluation $f(\alpha)$ of $f(X)$. In this case, to secret share s, $f(X)$ is sampled uniformly at random from the set $\mathbb{F}[X]_{\leq p}$ of polynomials of degree at most p, under the condition that $f(\alpha) = s$. The shares then correspond to n evaluations of $f(X)$ in points $\alpha_1, \ldots, \alpha_n \in \mathbb{F} \setminus \{\alpha\}$. This variant has exactly the same properties as before.

Furthermore, Shamir's scheme can easily be adjusted to accommodate secrets of larger dimension m. In this packed secret-sharing variant, to share a vector $\mathbf{x} \in \mathbb{F}^m$, the polynomial $f(X)$ is sampled uniformly at random from the set $\mathbb{F}[X]_{\leq m+p-1}$ of polynomials of degree at most $m + p - 1$, under the condition that $\big(f(1), \ldots, f(m)\big) = \mathbf{x}$. The secret shares correspond to n evaluations of $f(X)$ in pairwise distinct points $\alpha_1, \ldots, \alpha_n \in \mathbb{F} \setminus \{1, \ldots, m\}$, where we assume that $|\mathbb{F}| \geq n + m$. Shamir's packed secret-sharing scheme for sharing m-dimensional vectors $\mathbf{x} \in \mathbb{F}^m$, instantiated with privacy parameter p, has $(m+p)$-reconstruction, p-privacy and $(2m + 2p - 1)$-product-reconstruction. In particular, player subsets of cardinality k, with $p < k < m + p$, are neither qualified nor unqualified.