New Tools for Multi-Party Computation

Lisa Kohl
lisa.kohl@kit.edu

Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
Karlsruhe Institute of Technology, Karlsruhe, Germany

This work is the Master's thesis of Lisa Kohl. It was developed in the research group Cryptology at the Centrum Wiskunde & Informatica under the supervision of Ronald Cramer in cooperation with the research group Cryptography and IT Security at the Karlsruhe Institute of Technology under the supervision of Dennis Hofheinz.

Abstract. In this work we extend the electronic voting scheme introduced by R. Cramer, R. Gennaro and B. Schoenmakers in [CGS97]. In the original paper the privacy of votes is based on the decisional Diffie-Hellman or respectively the higher residuosity assumption. Since both problems can be solved efficiently in the event of quantum computers, a desirable goal is to implement the voting scheme with privacy based on different assumptions. We present the framework and a concrete instantiation for an efficient solution with privacy based on learning with errors over rings. Additionally we show how to achieve privacy assuming hardness of worst-case lattice problems, which are well analyzed and conjectured to be secure against quantum computers.

Introduction

By the results of P. Shor problems like decisional Diffie-Hellman or the higher residuosity problem are solvable in polynomial time on quantum computers [Sho97]. Even though quantum computers are yet far from being practical, it is important to develop cryptographic constructions based on alternative assumptions. This is particularly true for constructions like electronic voting schemes, as it is desirable to preserve the privacy of votes into the future.

One promising approach for post-quantum security is lattice-based cryptography. For $n \in \mathbb{N}$ a *lattice* is a discrete additive subgroup of the vector space $\mathbb{R}^n$. An example for a basic lattice-problem is the so-called *shortest vector problem* SVP, which is finding the shortest nonzero vector in a given lattice. It is conjectured to be hard to approximate SVP within polynomial factors even with a quantum-computer [MR08].

Another problem is given natural numbers $n, q \in \mathbb{N}$ and an error distribution χ to determine the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ given tuples of the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ for $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly random and $e \leftarrow \chi$ on $\mathbb{Z}_q$. This so-called *learning with errors* (LWE) problem seems to be completely unrelated on a first glance. In 2009 though O. Regev gave in [Reg09] a quantum reduction from worst-case lattice problems to the learning with errors problem, later the year a classical reduction was provided by C. Peikert in [Pei09]. Since then a lot of cryptographic constructions have been built on top.

A variant of LWE is the so-called *learning with errors over rings* (RLWE). The ring considered for this problem is of the form $\mathbb{Z}_q[X]/f$ for a polynomial $f \in \mathbb{Z}_q[X]$ instead of merely $\mathbb{Z}_q$. The additional algebraic structure allows for more efficient constructions. As shown by V. Lyubashevsky, C. Peikert and O. Regev in [LPR13] learning with errors over rings is provably as secure as worst-case problems on a special group of lattices, the so-called ideal lattices.

The electronic voting scheme proposed in [CGS97] is very efficient in terms of time and communication complexity on the side of both the voters and the tallying authorities. It provides security properties as privacy, robustness and universal verifiability.

The idea for casting votes is to let each voter publishes an encryption of his vote supplemented by a proof that the ciphertext indeed is the encryption of a valid vote without leaking knowledge about the vote itself. The invalid votes are discarded and the remaining ciphertexts added up. Finally the sum of the ciphertexts is commonly decrypted by the tallying authorities to obtain the result of the election.

We will present different instantiations of the underlying cryptosystem. The obtained electronic voting schemes will provide eligibility, privacy, robustness, copy protection and partial universal verifiability.

For this purpose the cryptosystem has to comply with several properties. First it has to support a proof of partial knowledge or *OR-proof* as broached by R. Cramer, I. Damgård and B. Schoenmakers in [CDS94]. We generalize the results of this paper and show how to obtain OR-proofs for the class of semi-homomorphic encryption schemes as introduced by R. Bendlin et al. in [BDOZ11].

The second major requirement on the underlying public key encryption system is that a sum of ciphertexts decrypts to the sum of the corresponding plaintexts. A cryptosystem complying with this requirement is called additive homomorphic. We relax this notion and only require the system to be roughly additive homomorphic. This means we require the cryptosystem to be additive homomorphic only in case the number of the added ciphertexts does not exceed a certain bound.

To enable mutual decryption the encryption scheme must support distributed decryption, where the secret key is shared among several authorities. To obtain privacy even in the event of corrupted authorities, we require the scheme to be threshold semantic secure. This roughly means that a ciphertext does not leak any information about the encrypted plaintext to any coalition of corrupted authorities of size less than a certain threshold.

We employ and extend the methods presented by R. Bendlin and I. Damgård in [BD10] to obtain protocols for threshold decryption for the presented cryptosystems. Furthermore we use and adapt the protocols for secure multi-party computation presented in [BDOZ11] and by I. Damgård et al. in [DPSZ12] for achieving active security during decryption.

Throughout this work we require the modulus q to be super-polynomial in the security parameter. This enables statistically hiding error terms and thereby preventing leakage of information, for instance about the secret key during decryption.

For LWE there exist security reductions from the search version to the decision version for certain superpolynomial moduli ([Pei09], [ACPS09], [MM11]). Unfortunately such a reduction proof does not yet exist for learning with errors over rings with super-polynomial modulus. Thus only for cryptosystems based on the former the security can be directly based on the hardness of worst-case lattice problems.

We start this work by introducing basic terms and concepts in Chapter 1. We give an overview of the electronic voting scheme of [CGS97] as framework and state the requirements on the underlying public key encryption scheme in Chapter 2. In Chapter 3 we consider instantiating the election scheme with cryptosystems based on learning with errors over rings and give an overview of the NTRU encryption scheme introduced in [HPS98] as concrete instantiation. We show how to achieve security against active adversaries during decryption in this setting with the protocols for multi-party computation presented in [BDOZ11] and [DPSZ12] in Chapter 4. In Chapter 5 we give an overview of the cryptosystems presented by Applebaum et al. in [ACPS09] and by O. Regev in [Reg05] respectively. We show that they provide an electronic voting scheme with privacy based on worst-case lattice problems. Finally we give an overview of the efficiencies of the different instantiations compared to the original scheme in Chapter 6.

Contents

1 Notation and Preliminaries

Throughout this work we will consider two security parameters, one computational security parameter $\kappa \in \mathbb{N}$ and one statistical parameter $u \in \mathbb{N}$. In practice the latter can be chosen much smaller than κ, as it does not depend on the computational power of the adversary. Since we consider security in the asymptotic sense we will view all parameters ρ used throughout the work as functions $\rho\colon \mathbb{N} \to S_\rho$ for some set S_ρ. We write $\rho := \rho(\kappa) \in S_\rho$ to define a function $\rho\colon \mathbb{N} \to S_\rho$ and furthermore to imply that by ρ in the following we also denote the element $\rho(\kappa) \in S_\rho$ for fixed κ. For $\rho_1 := \rho_1(\kappa) \in S_{\rho_1}$ and $\rho_2 := \rho_2(\kappa) \in S_{\rho_2}$ we write $\rho_1 < \rho_2$ to require $\rho_1(k) < \rho_2(k)$ for all $k \in \mathbb{N}$ and accordingly for other relational operators.

Let $f \in \mathbb{Z}[X]$ be a polynomial and let $n \in \mathbb{N}$ and $f_1, \ldots, f_n \in \mathbb{Z}$ such that $f = \sum_{i=1}^n f_i X^{i-1}$. Then we let f additionally denote the function $f\colon \mathbb{Z} \to \mathbb{Z},\ z \mapsto \sum_{i=1}^n f_i z^{i-1}$ determined by the polynomial f.

Further by requiring ρ to be polynomially bounded, we demand the existence of a polynomial $f \in \mathbb{Z}[X]$ and a natural number $k_0 \in \mathbb{N}$ such that for all $k \in \mathbb{N}$ with $k > k_0$ we have $\rho(k) \leq f(k)$.

For each $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$ of the first n natural numbers and by $\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z}$ the congruence classes of integers modulo n. For n a prime power we denote by $\mathbb{F}_n$ the finite field with n elements, which is unique upto isomorphism. Note that $\mathbb{F}_n = \mathbb{Z}_n$ if and only if n is prime.

To refer to publications this work is based on we use two different notations. For results taken verbatim or with unmodified content we give the citation in brackets or optionally with the keyword "see". If we use underlying ideas but translate them to a different context, we refer to the original work with the keyword "compare".

1.1 Norm and Distribution

In this section we introduce terms and notations used throughout the work, based on Section 4 of [Lyu12] and the preliminaries of [ACPS09] and [BD10]. We start by defining the standard Euclidean and infinity norm.

Definition 1.1 (Absolute Value and Norm). Let $q \in \mathbb{N}$ be an natural number. For any element $z \in \mathbb{Z}_q$ we denote with $|z|$ the absolute value of the representative of z in the interval $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$.

Let $Z \in \{\mathbb{Z}, \mathbb{Z}_q\}$. By $\|\cdot\|\colon Z^n \to \mathbb{R}_{\geq 0},\ (x_i)_{i\in[n]} \mapsto \sqrt{\sum_{i=1}^n |x_i|^2}$ we denote the *Euclidean norm* on Z^n and by $\|\cdot\|_\infty\colon Z^n \to \mathbb{N},\ (x_i)_{i\in[n]} \mapsto \max(|x_1|, \ldots, |x_n|)$ the *infinity norm* on Z^n.

We will work with the Gaussian distribution or normal distribution as error distribution of the learning with errors problem. For a distribution λ with density function D by $z \leftarrow \lambda$ or $z \leftarrow D$ we denote drawing an element according to distribution λ.

Definition 1.2 (Gaussian distribution). For $r \in \mathbb{R}_{>0}$ by ψ_r we denote the *continuous Gaussian distribution* on $\mathbb{R}$ with density function $D_r \colon \mathbb{R} \to \mathbb{R},\ x \mapsto 1/r \cdot \exp(-\pi\,(x/r)^2)$. The distribution ψ_r has expected value $\mu = 0$ and standard deviation $\sigma = r/\sqrt{2\pi}$.

Let $q \in \mathbb{N}$. Then the *discrete Gaussian distribution* $\overline{\psi}_{q,r}$ on $\mathbb{Z}_q$ is obtained by drawing $y \leftarrow D_{r'}$ and returning $\lfloor q \cdot y \rceil \mod q$. With $\overline{\psi}^k_{q,r}$ we denote the distribution on $\mathbb{Z}_q^k$ obtained by drawing each component of the vector according to $\overline{\psi}_r$. We sometimes simply write $\overline{\psi}_r$, when the modulus q is clear without ambiguity.

Definition 1.3 (Discrete Gaussian distribution on $\mathbb{Z}^n$). Let $\sigma \in \mathbb{R}$, $\mathbf{v} \in \mathbb{Z}^n$ and let further $\rho^n_{\mathbf{v},\sigma} \colon \mathbb{R}^n \to \mathbb{R} \colon \mathbf{x} \mapsto 1/\sqrt{2\pi} \cdot \exp(-\|\mathbf{x}-\mathbf{v}\|^2/(2\sigma^2))$ be the density function corresponding to the continuous Gaussian distribution on $\mathbb{R}^n$. We define the *discrete Gaussian distribution over the integer lattice* $\mathbb{Z}^n$ with center $\mathbf{v}$ and standard deviation σ by its density function $D^n_{\mathbf{v},\sigma} \colon \mathbb{Z}^n \to \mathbb{R}, \mathbf{x} \mapsto \rho^n_{\mathbf{v},\sigma}(\mathbf{x})/\rho_\sigma(\mathbb{Z}^n)$, where $\rho_\sigma(\mathbb{Z}^n) := \sum_{\mathbf{z}\in\mathbb{Z}^n} \rho^n_{\mathbf{0},\sigma}(\mathbf{z})$. For the distribution centered at $\mathbf{0} \in \mathbb{Z}^n$ we define $D^n_\sigma := D^n_{\mathbf{0},\sigma}$.

The following two lemmas will be of importance for bounding the error terms in ciphertexts and enable us to prove correctness of the respective encryption schemes. We refer to [Lyu12] for the proof.

Lemma 1.4 ([Lyu12] Lemma 4.3). *For any $n \in \mathbb{N}$, $\mathbf{v} \in \mathbb{Z}^n$ and $\sigma, t \in \mathbb{R}_{>0}$ we have*

$$\Pr\left[|\langle \mathbf{z}, \mathbf{v}\rangle| > t \mid \mathbf{z} \leftarrow D^n_\sigma\right] \le 2e^{-\frac{t^2}{2\|\mathbf{v}\|^2\sigma^2}}.$$

Lemma 1.5 ([Lyu12] Lemma 4.4). *Let $n \in \mathbb{N}$ and $\sigma, t \in \mathbb{R}_{>0}$. Then*

$$\Pr\left[\|\mathbf{z}\|_\infty > t\sigma \mid \mathbf{z} \leftarrow D^n_\sigma\right] \le 2e^{-\frac{t^2}{2}}$$

and for $t > 1$ we have

$$\Pr\left[\|\mathbf{z}\| > t\sigma\sqrt{n} \mid \mathbf{z} \leftarrow D^n_\sigma\right] \le t^n \cdot 2e^{\frac{n(1-t^2)}{2}}.$$

The following definitions on distribution ensembles are based on Section 4.1 [CDN01]. Let D be an arbitrary domain, e.g. the domain of bit strings of arbitrary length. For each $k \in \mathbb{N}$ and $x \in D$ we denote with $X(k,x)$ a random variable. We then call the infinite family $X := \{X(k,x)\}_{k\in\mathbb{N},x\in D}$ a *probability distribution ensemble indexed by D.*

Definition 1.6 (Statistical Indistinguishability). Let X and Y be two distribution ensembles indexed with D. We say X and Y are *statistically indistinguishable* if there exists a negligible function $\mathrm{negl} \colon \mathbb{N} \to [0,1]$ and a $k_0 \in \mathbb{N}$ such that for all $k > k_0$ and all $x \in D$ it holds

$$\frac{1}{2}\sum_{y\in\mathbb{R}} \left|\Pr[X(k,x) = y] - \Pr[Y(k,x) = y]\right| < \mathrm{negl}(k).$$

In this case we write $X \approx_s Y$.

Definition 1.7 (Computational Indistinguishability)**.** Let X and Y be two distribution ensembles indexed by D. We say X and Y are *computationally indistinguishable* if there exists a negligible function $\operatorname{negl}\colon \mathbb{N} \to [0,1]$ such that for every PPT TM $\mathcal{A}$ there exists a $k_0 \in \mathbb{N}$ such that for all $k > k_0$ and all $x \in D$ and $w \in \mathbb{R}$ we have

$$\left|\Pr[\mathcal{D}(1^k, x, w, X(k,a)) = 1] - \Pr[\mathcal{D}(1^k, x, w, Y(k,a)) = 1]\right| < \operatorname{negl}(k).$$

In this case we write $X \approx_c Y$.

An important idea is to hide error terms by adding a value randomly drawn from a interval super-polynomial larger than the interval the error terms are from. This technique is called “smudging”.

Lemma 1.8 ([AJL+12] Lemma 1, Smudging)**.** *Let κ be the security parameter and let $\operatorname{negl}\colon \mathbb{N} \to \mathbb{R}_{>0}$ be a negligible function. Let $b_1 := b_1(\kappa), b_2 := b_2(\kappa) \in \mathbb{N}$ be bounds with $b_1(\kappa)/b_2(\kappa) \leq \operatorname{negl}(\kappa)$. Let $e := e(\kappa) \in [-b_1, b_1]$ be an arbitrary integer and $\psi := \psi(\kappa)$ be the uniform distribution on $[-b_2, b_2] \cap \mathbb{Z}$. Then the distribution $e + \psi$, obtained by drawing an $\tilde{e} \in \psi$ and returning $e + \tilde{e}$, is statistically indistinguishable to the distribution ψ.*

Proof. Without loss of generality let $e \geq 0$ and

$$\Delta(y) := |\Pr[e + \tilde{e} = y \mid \tilde{e} \leftarrow \psi] - \Pr[\tilde{e} = y \mid \tilde{e} \leftarrow \psi]|.$$

We have

$$\begin{aligned}
\frac{1}{2}\sum_{y\in\mathbb{Z}} \Delta(y) &= \frac{1}{2}\left(\sum_{y\in[-b_2,-b_2+e)} \Delta(y) + \sum_{y\in[-b_2+e,b_2]} \Delta(y) + \sum_{y\in(b_2,b_2+e]} \Delta(y)\right)\\
&= \frac{1}{2}\left(\sum_{y\in[-b_2,-b_2+e)} \frac{1}{2b_2+1} + 0 + \sum_{y\in(b_2,b_2+e]} \frac{1}{2b_2+1}\right)\\
&= \frac{1}{2}\left(2\frac{e}{2b_2+1}\right) \leq \frac{b_1}{2b_2+1} \leq \operatorname{negl}(\kappa),
\end{aligned}$$

which shows that the two distributions are statistically indistinguishable. □

Alternatively the translation of a Gaussian distribution can be hidden by discarding the output with a certain probability - this technique is called “rejection sampling”. We cite the following theorem without proof from [Lyu12].

Theorem 1.9 ([Lyu12] Theorem 4.9, Rejection Sampling)**.** *Let $n, T \in \mathbb{N}$ be natural numbers and $U \subseteq \mathbb{Z}^n$ a subspace of $\mathbb{Z}^n$, such that all elements in U have norm less than T. Let further $D\colon U \to \mathbb{R}$ be a probability distribution and $\sigma \in \omega\left(T\sqrt{\log n}\right)$. Then there exists a constant $M \in O(1)$ such that the output distributions of the algorithm*

A_1: draw $v \leftarrow D$, $z \leftarrow D^n_{v,\sigma}$ and output (z, v) with probability $D^n_\sigma(z)/(MD^n_{v,\sigma})$
and

A_2: draw $v \leftarrow D$, $z \leftarrow D^n_\sigma$ and output (z, v) with probability $1/M$
have at most statistical distance $2^{-\omega(\log n)}/M$. In particular A_1 outputs something with probability at least $1 - 2^{-\omega(\log n)}/M$.

For a concrete instantiation $\sigma = \alpha T$ for $\alpha \in \mathbb{R}_{>0}$ we have $M = e^{12/\alpha+1/(2\alpha^2)}$ and the outputs of A_1 and A_2 are within statistical distance $2^{-100}/M$.

1.2 Learning with Errors

As brought up in the introduction the learning with errors problem will be of special importance for this work. After introducing the search and the decision variant (compare [ACPS09], [AJL+12] and [BD10]) we cite three lemmas. The first two give us the search to decision reduction for certain possibly super-polynomial moduli. The third allows us to transform the problem to the so-called "Hermite normal form". Finally we present the main theorem of [Pei09] covering the reduction from the lattice problem GapSVP to the search version of the learning with errors problem.

The search version of the learning with errors problem is to recover a secret vector given polynomially many noisy random inner products.

Definition 1.10 (Learning with Errors (Search Problem)). Let $n := n(\kappa) \in \mathbb{N}$ and $q := q(\kappa) \in \mathbb{N}$ with $q \geq 2$ and $\lambda := \lambda(\kappa)$ an error distribution over $\mathbb{Z}_q$. For $\mathbf{s} \in \mathbb{Z}_q^n$ by $A_{\mathbf{s},\lambda}$ we denote the distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ which is obtained by drawing a vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly random and an error term e according to λ and then outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The *learning with errors search problem* $\mathrm{LWE}_{q,\lambda}$ in n dimensions is to solve for $\mathbf{s} \in \mathbb{Z}_q^n$ with noticeable probability given independent samples from $A_{\mathbf{s},\lambda}$.

$\mathrm{LWE}_{q,\lambda}$ is said to be *hard*, if there is no probabilistic algorithm running in polynomial time solving the problem for infinitely many n.

The goal of the decision version is to distinguish between the distribution of noisy random inner products and uniform distribution.

Definition 1.11 (Learning with Errors (Decision Problem)). Let all parameters be defined as in the previous definition. Let O be an oracle that either draws a vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly random and produces samples according to $A_{\mathbf{s},\lambda}$ or produces samples according to uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The *learning with errors decision problem* $\mathrm{DLWE}_{q,\lambda}$ is given access to O to distinguish between the two distributions with non-negligible advantage.

The search and decision variants are equivalent for certain choices of parameters, as shown by the following lemmas. For the proofs we refert to [ACPS09] and [Pei09] respectively.

Lemma 1.12 (Search-to-Decision-Reduction ([ACPS09], Lemma 1)). *Let $n := n(\kappa) \in \mathbb{N}$, $p := p(\kappa) \in \mathbb{P}$ and $\delta := \delta(\kappa) \in \mathbb{N}$ all be polynomially bounded and $q := q(\kappa) := p^\delta \in \mathbb{N}$ a prime power. Let $\lambda := \lambda(\kappa)$ be an error distribution over $\mathbb{Z}_q$ that produces elements with absolute norms bounded by $\frac{p-1}{2}$ with overwhelming probability. Then there is a reduction in probabilistic polynomial time from $\mathrm{LWE}_{q,\lambda}$ to $\mathrm{DLWE}_{q,\lambda}$.*

Lemma 1.13 ([Pei09], Lemma 3.6). *Let $n := n(\kappa) \in \mathbb{N}$ and $\delta := \delta(\kappa) \in \mathbb{N}$ polynomially bounded. Let $\alpha := \alpha(\kappa) \in (0,1)$. For each $i \in [\delta]$ let $p_i := p_i(\kappa) \in \mathbb{N}$ polynomially bounded, distinct and such that there exists a $T \in \omega\left(\sqrt{\log n}\right)$ with $q_i \geq T/\alpha$. Let $q := q(\kappa) := \prod_{i=1}^{\delta(\kappa)} q_i(\kappa)$. Then there is a probabilistic polynomial time reduction from $\mathrm{LWE}_{q,\overline{\psi}_\alpha}$ to $\mathrm{DLWE}_{q,\overline{\psi}_\alpha}$.*

The following lemma proven in [ACPS09] allows us to transform the $\mathrm{DLWE}_{q,\lambda}$ problem to Hermite normal form, where the secret vector itself is chosen according to the error distribution λ.

Lemma 1.14 ([ACPS09], Lemma 2). *Let $n := n(\kappa) \in \mathbb{N}$, $p := p(\kappa) \in \mathbb{P}$ and $\delta := \delta(\kappa) \in \mathbb{N}$ all be polynomially bounded and $q := p^{\delta} \in \mathbb{N}$ a prime power. Let λ be an error distribution over $\mathbb{Z}_q$, $\mathbf{s} \in \mathbb{Z}_q$ arbitrary and $\mathbf{x} \leftarrow \lambda$. Then there exists a deterministic transformation T that maps $A_{\mathbf{s},\lambda}$ to $A_{\mathbf{x},\lambda}$ and the uniform distribution on $Z_q^n \times \mathbb{Z}_q$ to itself in polynomial time.*

As mentioned earlier we claimed that the hardness of LWE can be reduced to the hardness of worst-case lattice problems. In the following we first give a short introduction to lattices based on [Pei09] and [Reg09] and then state the main theorem of the former.

Let $n \in \mathbb{N}$. An lattice is an additive discrete subgroup of $\mathbb{R}^n$, in other words a set $\Lambda \subseteq \mathbb{R}^n$ is called a *lattice*, if there exists a set $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subseteq \mathbb{R}^n$ of n linearly independent vectors such that

$$\Lambda := \{\sum_{i=1}^{n} z_i \cdot \mathbf{b}_i \mid \forall i \in [n] \colon z_i \in \mathbb{Z}\}.$$

The set $\mathcal{B} := \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is called a *basis* of Λ.

The *minimum distance* $\lambda_1(\Lambda)$ of the lattice Λ is defined as the length of the shortest non-zero vector in Λ, that is

$$\lambda_1(\Lambda) := \min_{0 \neq \mathbf{v} \in \Lambda} \|v\|.$$

The following problem is the decision variant of approximating the shortest vector in a given lattice.

Definition 1.15 (GapSVP). Let $\gamma := \gamma(\kappa) \in \mathbb{R}$ be a function with $1 \leq \gamma$. Let Λ be a κ-dimensional lattice and $d \in \mathbb{R}_{>0}$ a positive real number. Then the *shortest vector problem* GapSVP_{γ} is to decide whether (Λ, d) is a YES instance, that is $\lambda_1(\Lambda) \leq d$, or a NO instance, that is $\lambda_1(\Lambda) > \gamma \cdot d$.

An alleviated version of GapSVP was introduced in [Pei09].

Definition 1.16 (ζ-to-γ-GapSVP). Let $\zeta := \zeta(\kappa), \gamma := \gamma(\kappa) \in \mathbb{R}$ be such that $1 \leq \lambda \leq \gamma$. Let Λ be a κ-dimensional lattice with $1 \leq \lambda_1(\Lambda) \leq \zeta$ and $d \in \mathbb{R}$ be a real number with $1 \leq d \leq \zeta/\gamma$. Then the *$\zeta$-to-$\gamma$ shortest vector problem* $\mathrm{GapSVP}_{\zeta,\gamma}$ is to decide whether the input pair (Λ, d) is a YES instance, that is $\lambda_1(\Lambda) \leq d$, or a NO instance, that is $\lambda_1(\Lambda) > \gamma \cdot d$.

In case $\zeta(k) \geq 2^{k/2}$ for all $k \in \mathbb{N}$ the shortest vector problem GapSVP_{γ} and the ζ-to-γ shortest vector problem $\mathrm{GapSVP}_{\zeta,\gamma}$ are equivalent. The former is conjectured to be hard even in the event of quantum-computers [MR08].

Furthermore $\mathrm{GapSVP}_{\zeta,\gamma}$ is conjectured to be hard in the worst case when $\zeta > 2\gamma$, or in other words when q is chosen such that there exists an $r := r(\kappa) \in \omega\left(\sqrt{n(\kappa)}\right)$ with $q > 2 \cdot r/\alpha$ [Pei09].

The following theorem of [Pei09] provides the first classical reduction from a worst-case lattice problem to the search version of the learning with errors problem.

Theorem 1.17 ([Pei09], Theorem 3.1). *Let $n := n(\kappa) \in \mathbb{N}$ polynomially bounded by κ, $\lambda := \lambda(\kappa) \in \mathbb{R}$ such that $\lambda \geq n/(\alpha\sqrt{\log n})$. Let further $\zeta := \zeta(\kappa) \in \mathbb{R}$ be such that $\zeta \geq \lambda$ and $q := q(\kappa) \in \mathbb{N}$ such that there exists $t := t(\kappa) \in \omega(\sqrt{\log n/n})$ such that $q \geq \zeta \cdot t$. Then there exists a probabilistic polynomial time reduction from solving the shortest vector problem* $\mathrm{GapSVP}_{\zeta,\gamma}$ *with overwhelming probability in the worst case to solving* $\mathrm{LWE}_{q,\overline{\psi}_\alpha}$ *using polynomially many samples.*

Altogether this will later allow us to base the security of the presented cryptosystems on the hardness of worst-case lattice problems.

1.3 Learning with Errors over Rings

The learning with errors over rings problem (RLWE) is an algebraic variant of the *opLWE* problem introduced in the previous section which allows for more efficient applications. Before giving the definition we provide some algebraic basics.

Remark 1.18 (Cyclotomic Polynomial). Let $N \in \mathbb{N}$ and $n := \varphi(N)$, where

$$\varphi\colon \mathbb{N} \to \mathbb{N},\ n \mapsto |\mathbb{Z}_n^\times|$$

denotes Euler's phi function. A number $\zeta \in \mathbb{C}$ is called a *primitive N-th root of unity* if $\zeta^N = 1$ and for all $k \in \mathbb{N}$ with $k < N$ it holds $\zeta^k \neq 1$. Note that for all $j \in [n]$ the value $\zeta_N^j := \exp(2\pi ji/N) \in \mathbb{C}$ is a primitive N-th root of unity.

The *N-th cyclotomic polynomial* $\Phi_N \in \mathbb{Z}[X]$ is the unique irreducible polynomial of degree n which factors to

$$\Phi_N = \prod_{j \in \mathbb{Z}_N^\times} (X - \zeta_N^j)$$

over $\mathbb{C}$.

Note that in case N is a power of 2 the N-th cyclotomic polynomial is of the form $\phi_N = X^n + 1$ for $n = N/2$.

Let $N, P \in \mathbb{N}$, $n := \varphi(n)$, ϕ_N the N-th cyclotomic polynomial and $R := \mathbb{F}_P[X]/\phi_N$. Then for each $f \in R$ there exist unique $f_1, \dots f_n \in \mathbb{F}_P$ such that $f = \sum_{l=1}^n f_l X^{l-1}$. We consider the induced surjective coefficient embedding

$$\iota\colon R \to \mathbb{F}_P^n,\ \sum_{l=1}^n f_l X^{l-1} \mapsto (f_l)_{l\in[n]}.$$

Using this bijection we can straightforward transfer norm and Gaussian distribution introduced in the first section to R. For $f \in \mathbb{R}$ we define $\|f\|_\infty := \|\iota(f)\|_\infty$ and $\|f\| := \|\iota(f)\|$.

Further for $\alpha \in \mathbb{R}_{>0}$ and $g \in R$ by $D_{g,\alpha}^n$ or simply $D_{g,\alpha}$ we denote the distribution obtained by drawing a vector $\mathbf{z}$ according to the discrete Gaussian distribution over the integer lattice $\mathbb{Z}^n$ with center $\iota(g)$ and returning $\iota^{-1}(\mathbf{z} \bmod P)$. We define $D_\alpha := D_{0,\alpha}$.

The following lemma from [BCK+14] guarantees that in case N is a power of 2 certain polynomials in $\mathbb{Z}[X]/\phi_N$ have inverses with small coefficients. This property is necessary for the efficient proof of knowledge established in [BCK+14] and adapted in Section 3.2.1 for proving validity of votes encrypted with the so-called NTRU cryptosystem.

Lemma 1.19 ([BCK+14], Lemma 3.1)**.** *Let $n \in \mathbb{N}$ be a power of 2. Then for all $i, j \in \{0, \dots, 2n-1\}$ the element $2(X^i - X^j)^{-1} \in \mathbb{Z}[X]/(X^n+1)$ has only coefficients in $\{-1, 0, 1\}$.*

For proving correctness of the NTRU cryptosystem we will need the estimates established in the coming lemma.

Lemma 1.20. *Let $n \in \mathbb{N}$ be a natural number, $P \in \mathbb{P}$ be a prime and $f, g \in \mathbb{F}_P[X]/(X^n+1)$ be polynomials. Then it holds*

$$\|f\|_\infty \leq \|f\| \leq \sqrt{n}\|f\|_\infty \text{ and } \|f \cdot g\|_\infty \leq n \cdot \|f\|_\infty \cdot \|g\|_\infty.$$

Proof. Let $R_P := \mathbb{F}_P[X]/(X^n+1)$. Let f_i, g_i for $i \in [n]$ the coefficients of f and g respectively, that is such that $f = \sum_{i=1}^n f_i X^{i-1} \in R_P$ and $g = \sum_{i=1}^n g_i X^{i-1} \in R_P$. Let $f_{\max} \in \mathbb{F}_P$ be the coefficient of f with maximal absolute value and $g_{\max} \in \mathbb{F}_P$ respectively. Then we have the following series of inequalities proving the first part

$$\|f\|_\infty = |f_{\max}| \leq \sqrt{\sum_{i=1}^n |f_i|^2} = \|f\| \leq \sqrt{\sum_{i=1}^n |f_{\max}|^2} = \sqrt{n} \cdot \|f\|_\infty.$$

Further it holds

$$f \cdot g = \sum_{i=1}^n \left(\sum_{j=1}^i f_j g_{i-j+1} X^{i-1} - \sum_{j=i+1}^n f_j g_{n+i-j+1} X^{i-1} \right)$$

and thus

$$\|f \cdot g\|_\infty \leq \sum_{j=1}^n |f_{\max}| \cdot |g_{\max}| = n \cdot \|f\|_\infty \cdot \|g\|_\infty.$$

□

Similar to LWE for specific sets of parameters there exist reductions from RLWE to worst-case problems on ideal lattices (compare [LPR13]). Unfortunately for the choice of the modulus q super-polynomial in the security parameter, a reduction from the search problem to the decision variant of RLWE is not yet known, thus for our application we have to directly assume the hardness of the RLWE decision problem.

Definition 1.21 (Learning with Errors Over Rings (decisional))**.** Let $q \in \mathbb{N}$, $\alpha \in \mathbb{R}_{>0}$ and $\phi \in \mathbb{Z}[X]$ with degree $n \in \mathbb{N}$. Let $R_q := \mathbb{Z}_q[X]/\phi$ and $e \leftarrow R_q$ drawn uniformly random. Let $A_{e,\alpha}$ be the distribution on $R_q \times R_q$ obtained by drawing $a \leftarrow R_q$ and $e \leftarrow D_\alpha$ and returning $(a, ae+f)$. Then the decisional variant of the *learning with errors over rings problem* $\mathrm{RLWE}_{q,\alpha}^{\phi}$ is to distinguish between $A_{e,\alpha}$ and uniform distribution on $R_q \times R_q$.

As described in [SS11] instead of working with the ring-learning with errors problem directly we will consider a variant $\mathrm{RLWE}^{\times}_{\mathrm{HNF}}(\phi, q, \alpha)$. There we replace $A_{e,\alpha}$ by $A^{\times}_{e,\alpha}$, where a is drawn from $R_q^{\times}$ instead of R_q, and thus instead of uniform distribution on $R_q \times R_q$ we take uniform distribution $R_q^{\times} \times R_q$. Furthermore instead of choosing e uniformly random we choose it according to the Gaussian distribution D_α, thus putting the distribution into *Hermite normal form.* The RLWE problem stays hard under the first change if a uniformly drawn element is invertible with more than negligible probability, which will be the case in our setting. The reduction of the latter works by a transformation of samples as it was the case for learning with errors, for more details consult [LPR13].

1.4 Secret Sharing

We first give a general overview about secret sharing schemes, explain additive sharing shortly and finally present Shamir sharing as it was introduced in [Sha79]. The section is based on [Sha79], [CDS94] and [CDI05].

Sharing the secret key between the tallying authorities will prevent groups not exceeding a certain size from gaining knowledge of single votes, while at the same time enabling the authorities to jointly recover the final result of the election.

Let $m := m(\kappa) \in \mathbb{N}$ and $S := \{S_k\}_{k \in \mathbb{N}}$ a family of sets. Then a secret sharing scheme on S is a method to share a secret $s \in S$ between m players, such that some subsets of players can reconstruct the value s with the help of their respective shares. In the following we call those subsets *qualified sets.* A secret sharing scheme is called *perfect,* if any non qualified subset of players cannot gain any information about s. The collection of all qualified sets is called *access structure.* The access structure of a perfect secret sharing scheme is *monotone*, that is if $A \subseteq [m]$ is qualified, then any set $B \subseteq [m]$ such that $A \subseteq B$ is qualified as well.

A set of shares is called *consistent* with s, if all qualified subsets can reconstruct the shared value to the same value $s \in S$.

A secret sharing scheme is called *ideal,* if the shares are of the same length as the secret.

Let $D(s)$ denote the joint probability distribution of the shares held by each of the m players. For any set $A \subseteq [m]$ of participants we let $D_A(s)$ denote the restriction of $D(s)$ to A, that is the probability distribution of the shares held by each player in A. If the secret sharing scheme is perfect, then for any non-qualified set $A \subseteq [m]$ the distribution $D_A(s)$ is independent of s and we write $D_A := D_A(s)$.

An important property of secret sharing schemes is the so-called *smoothness.* The following definition is taken from [CDS94].

Definition 1.22 ([CDS94], Smoothness). Let $\{S_k\}_{k \in \mathbb{N}}$ be a family of sets and for each $k \in \mathbb{N}$ let $\mathcal{S}(k)$ be a perfect sharing scheme on S_k with $m(k)$ participants, where $m := m(\kappa) \in \mathbb{N}$ is polynomially bounded. Then the family of secret sharing schemes is called *smooth,* if it satisfies the following requirements:

1. All shares generated in $\mathcal{S}(\kappa)$ are of length polynomial in κ.

2. Generation of shares and reconstruction of a secret can be done in time polynomial in κ.

3. Consistency of the set of all m shares with a secret $s \in S_\kappa$ can be tested in time polynomial in κ.

4. Given any non-qualified subset $A \subseteq [m(\kappa)]$ and any secret $s \in S_\kappa$, the set of shares can be completed to a full set of shares consistent with s and distributed according to $D(s)$. Furthermore this can be done efficiently, that is in time polynomial in κ.

5. Given any non-qualified subset $A \subseteq [m(\kappa)]$ the probability distribution D_A is such that shares for players in A are independent and chosen uniformly random.

Let $p := p(\kappa) \in \mathbb{N}, m := m(\kappa) \in \mathbb{N}$. In the following we consider sharing schemes on $\mathbb{Z}_p$. For each $j \in [m]$ we let P_j denote the j-th player.

A very simple sharing scheme is the so-called *additive sharing*, where for each $i \in [m]$ the party P_i holds a value $z_i \in \mathbb{Z}_p$ and the shared value z is recovered as $z = \sum_{i=1}^m z_i$. The only qualified set is $[m]$ and if the shares of all but one players are chosen uniformly random the scheme is smooth.

For $t := t(\kappa) \in \mathbb{N}$ with $t \leq n$ we consider the so-called *(t, m)-Shamir sharing*, where the qualified sets are all subsets $I \subseteq [m]$ of size at least t. From now on we assume p is prime with $p \geq m$. Note that it would suffice to require that the smallest prime divisor of p is greater than m, as this guarantees that nonzero elements with absolute value at most m are invertible in $\mathbb{Z}_p$.

First we need to consider *Lagrange interpolation*, which is a method for recovering a polynomial given tuples of points of the corresponding function. Let $I \subseteq [m]$ of size t and for all $j \in I$ be given values $y_j \in \mathbb{Z}_p$. Then there exists a unique polynomial $\rho \in \mathbb{Z}_p[X]$ of degree at most $t-1$ such that for all $j \in I$ it holds $\rho(j) = y_j$ [Fuh12]. It is defined by

$$\rho := \sum_{i \in I} y_i \prod_{j \in I, j \neq i} (X - j)(i - j)^{-1}.$$

Now we describe the generation and reconstruction of a Shamir shared value with the help of a trusted party $\mathcal{F}$ for distributing the shares.

Generation of a Shamir sharing: Let $z \in \mathbb{Z}_p$ the value to be shared.

1. The trusted party $\mathcal{F}$ chooses uniformly random a polynom $\rho \leftarrow \mathbb{Z}_p[X]$ of degree at most $t-1$ such that $\rho(0) = s$.
2. For each $j \in [m]$ the party $\mathcal{F}$ sends the value $s_j := \rho(j)$ to the player P_j.

Reconstruction of a Shamir shared value: Let $I \subseteq [m]$ be an index set of size at least t and $i \in I$ the index of the player who wants to recover the shared value with help of the other players in I.

1. For each $j \in I \backslash \{i\}$ the player P_j sends s_j to P_i.
2. P_i uses Lagrange interpolation to calculate a polynomial $\rho' \in \mathbb{Z}_p[X]$ of degree $t-1$ such that for all $j \in I$ it holds $\rho'(j) = s_j$.
3. P_i can recover the Shamir shared value by calculating $\rho'(0)$.

Let $s \in \mathbb{Z}_p$ and $\rho \in \mathbb{Z}_p[X]$ a polynomial of degree at most $t-1$ such that $\rho(0) = s$. Then for every subset $I \subseteq [m]$ of size at least t we say $(\rho(i))_{i \in I}$ is a (t,m)-*Shamir sharing* of s.

Let further $I \subseteq [m]$ and values $s_i \in \mathbb{Z}_p$ for all $i \in I$ given and $\rho \in \mathbb{Z}_p[X]$ a polynomial such that for all $i \in I$ it holds $\rho(i) = s_i$. Then we say ρ is *consistent* with $(i, s_i)_{i \in I}$.

The correctness of Shamir sharing follows directly from the properties of Lagrange Interpolation.

The perfectness can be seen by the following. Let $I \subseteq [m]$ be of size $t-1$. Given a set of values $(i, y_i)_{i \in I}$ by the properties of Lagrange interpolation for each choice $s \in \mathbb{Z}_p[X]$ there is exactly one polynomial $\rho \in \mathbb{Z}_p[X]$ such that $\rho(i) = y_i$ for all $i \in I$ and $\rho(0) = s$. Thus given only the shares of $t-1$ players no information can be derived about the Shamir shared value.

Another nice property of Shamir sharing is its linearity. Let $\rho, \rho' \in \mathbb{Z}_p[X]$ be polynomials of degree at most $t-1$ with $\rho(0) = s$ and $\rho'(0) = s'$ and $c \in \mathbb{Z}_p$ an arbitrary constant. Then $\rho + \rho'$ corresponds to an (t,m)-Shamir sharing of $s + s'$ and $c \cdot \rho$ to an (t,m)-Shamir sharing of $c \cdot s$.

In the following remark we give an idea on how to see the smoothness of a family of Shamir's secret sharing schemes.

Remark 1.23 (Smoothness). Let $\mathcal{S} := \{\mathcal{S}(\kappa)\}_{\kappa \in \mathbb{N}}$ be a family of Shamir's secret sharing schemes and all other variables defined as before. Then $\mathcal{S}$ is smooth. In the following we give a short overview of the proof outline. The perfectness was shown before and with the previous notes it is straightforward to show that the scheme also complies with the first three requirements. Given a non-qualified subset $A \subseteq [m]$ of size at most $t-1$ with corresponding set of shares $\{s_i\}_{i \in A}$ and any secret $s \in \mathcal{S}$, the set of shares can be completed to a full set of with s consistent shares and proper distribution by choosing a polynomial ρ uniformly random from the set

$$\left\{f \in \mathbb{Z}_p[X] \mid f \text{ has degree at most } t-1,\ f(0) = s,\ \forall i \in A\colon f(i) = s_i\right\}.$$

For the last claim let $s \in S$ be the secret to be shared and

$$\rho \leftarrow \left\{f \in \mathbb{Z}_p[X] \mid f \text{ has degree at most } t-1,\ f(0) = s\right\}$$

drawn uniformly random. To prove the claim we need to show that the values $\{f(i)\}_{i \in A}$ are distributed independent and uniform. Let for $i \in [t]$ the values $f_i \in \mathcal{A}$ be such that $f = \sum_{i=1}^{f} f_i X^{i-1}$. We have $f_1 = s$ and f_i distributed uniformly random for all $i \in [t] \setminus \{1\}$. As the values $\{f(i)\}_{i \in A}$ are linearly independent linear combinations of the $\{f_i\}_{i \in [t] \setminus \{1\}}$ and A is of size at most $t-1$, the shares held by players in A are distributed as required.

We will need the following remark for enabling the tallying authorities during decryption to jointly add a smudging term to the final result to prevent leaking information about the secret key.

Remark 1.24 ([CDI05], Sharing of a Sum of Values). Let $I \subseteq [m]$ of size t and let for each $j \in I$ the player P_j holds a value $s_j \in \mathbb{Z}_p$. Then the players can create a (t,m)-Shamir sharing of $s := \sum_{j \in I} s_j \in \mathbb{Z}_p$ in the following way:
Let for all $j \in I$ the polynomial $g_{j,I} \in \mathbb{Z}_p[X]$ be the distinct polynomial of degree $t-1$

with $g_{j,I}(0) = 1$ and $g_{j,I}(i) = 0$ for all $i \in I\backslash\{j\}$ and define $d_{j,I} := g_{j,I}(j)$. Then the values $(d_{j,I} \cdot s_j)_{j\in I}$ form a Shamir sharing of s. To see this, we define the polynomial $\rho := \sum_{j\in I} g_{j,I}(X) \cdot s_j \in \mathbb{Z}_p[X]$. We have $\rho(0) = \sum_{j\in I} g_{j,I}(0) \cdot s_j = s$ and for all $i \in I$ we have $\rho(i) = \sum_{j\in I} g_{j,I}(i) \cdot s_j = d_{j,I} \cdot s_j$ as desired and as I is of size t the polynomial ρ of degree at most $t - 1$ is uniquely determined.

As a major part of this work is based on learning with errors over rings, we want to give a generalization of Shamir sharing to certain polynomial rings.

Remark 1.25 (Sharing of Polynomials). Let $N \in \mathbb{N}$ and ϕ_N the N-th cyclotomic polynomial of degree $n := \varphi(N)$. Let $R := \mathbb{Z}_p[X]/\phi_N$.

Let $f \in R$ be a polynomial to be shared between $m \in \mathbb{N}$ players. We say $(f_{(i)})_{i\in[m]} \in R^m$ is a (t, m)-Shamir sharing of f if for all $j \in [n]$ the j-th coefficients of $(f_{(i)})_{i\in[m]}$ form a (t, m)-Shamir sharing of the j-th coefficient of f.

If $g \in R$ is public polynomial and $(f_{(i)})_{i\in[m]} \in R^m$ is a (t, m)-Shamir sharing of f, then as a consequence of the linearity of Shamir sharing the values $(g \cdot f_{(i)})_{i\in[m]}$ form a (t, m)-Shamir sharing of $g \cdot f$ and $(g + f_{(i)})_{i\in[m]}$ form a (t, m)-Shamir sharing of $g + f$. Similarly if we have a (t, m)-Shamir sharing $(g_{(i)})_{i\in[m]} \in R^m$ of g, then $(g_{(i)} + f_{(i)})_{i\in[m]}$ is a (t, m)-Shamir sharing of $g + f$. We do not use the more efficient so-called packed secret sharing (see [**FY92**]) here, as it does not support multiplication.

In the following we prove that indeed $(g \cdot f_{(i)})_{i\in[m]}$ forms a Shamir sharing of $g \cdot f$. For clarity we will do that for the special case that N is a power of 2 and thus $\phi_N = X^n + 1$. Let $g_j, f_j, f_{(i),j} \in \mathbb{F}_q$ for $j \in [n]$ the coefficients of g, f and $f_{(i)}$ respectively, i.e. such that $g = \sum_{j=1}^n g_j X^{j-1} \in R$, $f = \sum_{j=1}^n f_j X^{j-1} \in R$ and $f_{(i)} = \sum_{j=1}^n f_{(i),j} X^{j-1} \in R$ for all $i \in [m]$ and further for all $j \in [n]$ the values $(f_{(i),j})_{i\in[m]} \in \mathbb{F}_q^m$ form a Shamir sharing of f_j. Then we have for all $i \in [m]$:

$$g \cdot f_{(i)} = \sum_{j=1}^{n} \left(\sum_{k=1}^{j} g_k f_{(i),j-k+1} - \sum_{k=j+1}^{n} g_k f_{(i),n+j-k+1} \right) X^{j-1}.$$

By the linearity of Shamir sharing for each $j \in [n]$ the values

$$\left(\sum_{k=1}^{j} g_k f_{(i),j-k+1} - \sum_{k=j+1}^{n} g_k f_{(i),n+j-k+1} \right)_{i\in[m]}$$

form a (t, m)-Shamir sharing of

$$\sum_{k=1}^{j} g_k f_{j-k+1} - \sum_{k=j+1}^{n} g_k f_{n+j-k+1},$$

which is the j-th coefficient of $g \cdot f$ and thus proves the claim.

The last property of Shamir sharing we employ is that given enough values a Shamir shared value can be recovered even if some parties lie about their shares. This will enable achieving security against active adversaries corrupting a small number of parties without additional techniques in Section 5.1.2.

Remark 1.26 (Error Correction)**.** Let $(s_i)_{i\in[m]} \in \mathbb{Z}_p^m$ be a (t, m)-Shamir sharing of a value $s \in \mathbb{Z}_p$ and let $t - 1 < m/3$. Let $I \subseteq [m]$ of size at most $t - 1$ and for each $i \in I$ an arbitrary value $s'_i \in \mathbb{Z}_p$ be given, for each $i \in [m]\backslash I$ let $s'_i := s_i$. Then given the values $(s'_i)_{i\in[m]}$ the correct value s can be recovered without knowledge of the set I. This holds as there is exactly one polynomial f of degree at most $t - 1$ such that there exists a set $S \subseteq [m]$ of size at least $m - t + 1$ such that $f(i) = s'_i$ for all $i \in S$. Such a polynomial exists as at least $m - t + 1$ values belong to the original (t, m)-Shamir sharing. Now assume there is a second polynomial f' of degree at most $t - 1$ such that there exists a set $S' \subseteq [m]$ of size at least $m - t + 1$ such that $f'(i) = s'_i$ for all $i \in S'$. Then the intersection $D := S \cap S'$ has size at least $m - 2(t - 1) = m - 2t + 2 > 3t - 3 - 2t + 2 = t - 1$, implying that the polynomial $f - f'$ has at least t zeros. As both polynomials are of degree at most $t - 1$ this implies $f = f'$.

1.5 Universally Composable Security

The security notion of universally composable security was introduced in [Can01]. Let $m := m(\kappa) \in \mathbb{N}$ and $t := t(\kappa) \in \mathbb{N}$ such that $t \leq m$. Throughout this work we will consider the security of m-party protocols, where up to $t - 1$ players are corrupted by the adversary. The following is based on the lecture notes on universal composability of [Can04], the multi-party computation model of Cramer et al. [CDN01] and the appendix of the SPDZ-protocol [DPSZ12].

Real-Life Model Let Π be an m-party protocol executed by parties $P_1, \dots P_m$, where each party is an interactive Turing machine. We consider an adversary $\mathcal{A}$ corrupting up to $t - 1$ parties, which is *static* and *active*. The means that the adversary has to decide on the set of corrupted players in the beginning and is allowed to run arbitrary code on corrupted players. We will view the adversary $\mathcal{A}$ as part of the environment $\mathcal{Z}$, which represents all protocols in the system but Π and is modeled as another interactive Turing machine. The environment interacts with Π in the following way:

- The environment $\mathcal{Z}$ starts with some external input $z \in \{0, 1\}^\star$ and prepares inputs for the players.
- The players execute the protocol Π with the previously prepared inputs for the honest players and with the adversary $\mathcal{A}$ taking control over the actions of corrupted parties.
- The final results of all honest players are sent to $\mathcal{Z}$.
- $\mathcal{Z}$ outputs a bit $b \in \{0, 1\}$.

We define the binary random variable $\mathrm{View}_\Pi(\kappa, \mathcal{Z}(z))$ as the output of $\mathcal{Z}$ on internal input $z \in \{0, 1\}^\star$ and security parameter $\kappa \in \mathbb{N}$ after interacting with Π, where the used randomness is chosen uniformly random. We define the view of the environment $\mathcal{Z}$ interacting with Π as the ensemble of distributions $\mathrm{View}_\Pi(\mathcal{Z}) := \{\mathrm{View}_\Pi(\kappa, \mathcal{Z}(z))\}_{\kappa\in\mathbb{N}, z\in\{0,1\}^\star}$.

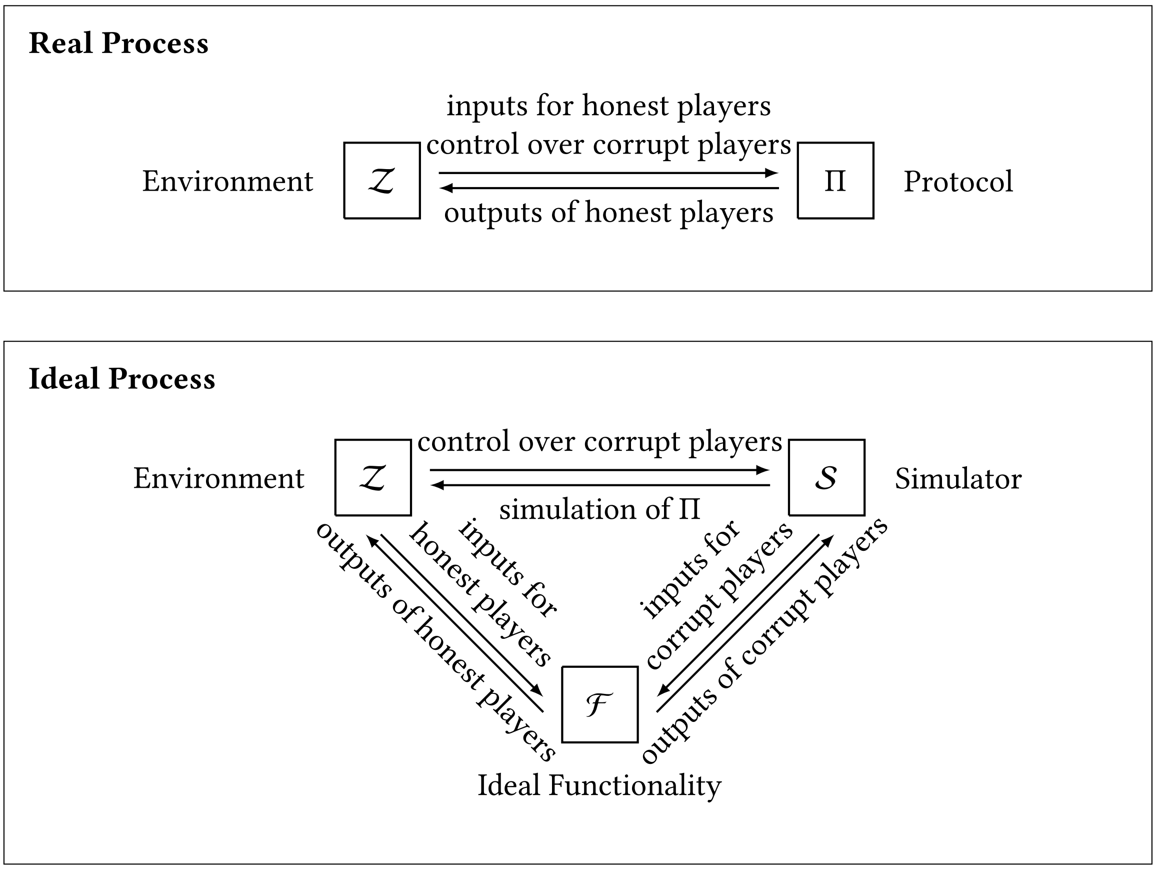

Figure 1.1: Real and Ideal Process (see [Dam15], Part 1, 40:40)

Ideal Model Security is defined by specifying the desired behavior of Π in an ideal functionality or trusted party $\mathcal{F}$ computing a function f and then proving that the environment cannot learn more from interacting with Π than it learns interacting with the ideal functionality. Therefore we have to show the existence of a simulator $\mathcal{S}$ that simulates the behavior of the corrupted parties towards the functionality and the view of the protocol Π towards the environment $\mathcal{Z}$. Here the parties $P_1, \ldots P_m$ are viewed as "dummy" parties. To be more precise $\mathcal{Z}$, $\mathcal{F}$ and $\mathcal{S}$ are interacting with each other in the following way:

- The environment $\mathcal{Z}$ starts with some external input $z \in \{0,1\}^\star$ and prepares inputs for the honest and corrupt players.
- $\mathcal{Z}$ gives the initial input of the corrupted players to the simulator $\mathcal{S}$ and the input for the honest players to the respective "dummy" players.
- Next the simulator $\mathcal{S}$ sends its input to the ideal functionality $\mathcal{F}$ and decides about the input the corrupted parties send to $\mathcal{F}$. The honest parties forward their input to the functionality $\mathcal{F}$.
- The ideal functionality evaluates the function f and sends output to the respective players and the simulator.

- Honest parties output the received value, while the output of corrupted players is decided by the simulator. The simulator and the players sent their outputs to the environment $\mathcal{Z}$.
- $\mathcal{Z}$ outputs a bit $b \in \{0,1\}$.

Now the random variable $\mathrm{View}_{\mathcal{F},\mathcal{S}}(\kappa, \mathcal{Z}(z))\}$ is defined as the output of $\mathcal{Z}$ on internal input $z \in \{0,1\}^\star$ and security parameter $\kappa \in \mathbb{N}$ after interacting with the simulator $\mathcal{S}$ on top of the ideal functionality $\mathcal{F}$, where again the used randomness is chosen uniformly random. We define the view of the environment $\mathcal{Z}$ interacting with $\mathcal{S}$ and $\mathcal{F}$ as the ensemble of distributions $\mathrm{View}_{\mathcal{F},\mathcal{S}}(\mathcal{Z}) := \{\mathrm{View}_{\mathcal{F},\mathcal{S}}(\kappa, \mathcal{Z}(z))\}_{\kappa\in\mathbb{N}, z\in\{0,1\}^\star}$.

Now we can define the notion of universally composable security.

Definition 1.27 (UC-Security (computational)). The protocol Π *UC-emulates* (or *implements*) the ideal functionality $\mathcal{F}$, if there exists a PPT simulator $\mathcal{S}$, such that for all environments $\mathcal{Z}$ it holds

$$\{\mathrm{View}_{\Pi}(\mathcal{Z})\} \approx_c \{\mathrm{View}_{\mathcal{F},\mathcal{S}}(\mathcal{Z})\}.$$

This means that the environment can not distinguish whether it is interacting with the real protocol Π or with the simulator $\mathcal{S}$ on top of the ideal functionality $\mathcal{F}$ (see Figure 1.1 for an intuition).

Hybrid Model Let $k \in \mathbb{N}$ and let $f_1, \ldots, f_k$ be m-party functions computable in probabilistic polynomial time and let $\mathcal{F}_1, \ldots, \mathcal{F}_k$ be ideal functionalities for evaluating $f_1, \ldots, f_k$ respectively. Then the execution of a protocol Π in the $(\mathcal{F}_1, \ldots, \mathcal{F}_k)$-hybrid model starts as in the real-life model, except that at predefined rounds the parties have access to the ideal functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_k$ to evaluate $f_1, \ldots, f_k$ as in the ideal model.

Finally we cite the central theorem of universally composable security, which allows to compose protocols while preserving security in the universal composable security model. For a proof we refer to [Can01].

Theorem 1.28 (Composition Theorem). *Let ρ be a protocol that UC-emulates an ideal functionality $\mathcal{F}$ and further $\Pi^{\mathcal{F}}$ an $\mathcal{F}$-hybrid protocol. Then the composed protocol $\Pi^{\mathcal{F}\to\rho}$ UC-emulates $\Pi^{\mathcal{F}}$, where $\Pi^{\mathcal{F}\to\rho}$ is the protocol obtained by replacing the calls to the ideal functionality $\mathcal{F}$ by executions of the protocol ρ.*

2 Electronic Voting

As brought up in the introduction this work is based on the election scheme introduced in [CGS97] by R. Cramer, R. Gennaro and B. Schoenmakers. In this chapter we will describe the approach of [CGS97] as framework to be instantiated with a suitable encryption scheme.

Let $M, m, t \in \mathbb{N}$. We will consider an (M, m, t)-electronic voting scheme, where M is the number of voters, m the number of authorities in charge of the tally and t such that we allow at most $t - 1$ of the authorities to be compromised. For the scheme to make sense we have the restriction $t \leq m$ and in typical applications we have $m \ll M$. A desirable goal is to minimize the work on the voters side, in particular voters should not be required to interact all with one another.

In the following we give an idea of the voting scheme presented in [CGS97], list desirable security properties, give an introduction to the underlying building blocks and finally present the framework for the secure electronic voting scheme of [CGS97] to be instantiated in Chapter 3 and 5.

For the sake of convenience we restrict to the case where a voter can vote yes or no by handing in either 1 or 0 as vote. Note that all building blocks can be extended to serve for an election with more than two options.

The voting scheme basically consists of two phases. We assume the existence of a public key encryption scheme with properties yet to be established. In the first phase each voter publish a ciphertext plus a proof that the ciphertext indeed is the encryption of a valid vote without leaking knowledge about the vote itself. For the release of the vote we assume the existence of a bulletin board.

Bulletin Board: A *bulletin board* is a public channel with memory for broadcasting messages. A section is assigned to each participant, where he can publish his messages. To prevent other participants from posting messages in this section publicly verifiable digital signatures are used. All messages are public and accessible to any party.

After the deadline of the election, each proof is checked by the authorities and the corresponding vote accepted or discarded respectively. The valid ciphertexts are added up and the tallying authorities jointly decrypt the sum of ciphertexts to obtain the final result. In Figure 2.1 we give an overview of the complete election procedure, where by $V_1, \ldots, V_M$ we denote the voters and by $A_1, \ldots, A_m$ the tallying authorities. Further for each $i \in [M]$ by c_i we denote the ciphertext and by Π_i the proof of validity provided by the voter V_i.

We consider the following aspects of security in this work. Note that eligibility directly follows from the properties of the bulletin board, where digital signatures prevent participants from posting messages to sections assigned to other participants. We will prove

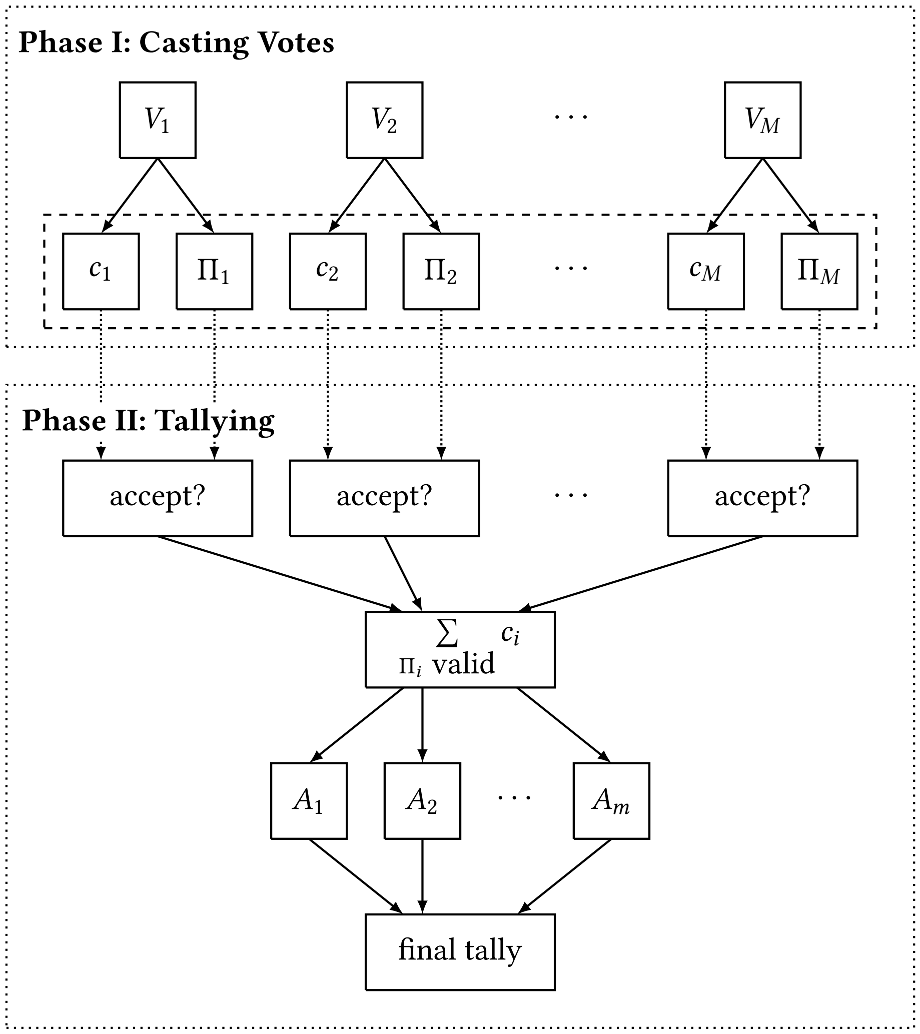

Figure 2.1: Overview of the Election Procedure

which properties of the underlying building blocks are necessary and sufficient to achieve the remaining security properties.

Privacy: An (M, m, t)-election scheme is called *private*, if an arbitrary set of at most $t - 1$ collaborating parties including tallying authorities can not gain any information about the value of an individual vote during or after the election process. We will consider *computational privacy*, that is privacy respective an underlying cryptographic assumption on the contrary to *information theoretic privacy.*

Eligibility: An electronic voting scheme is called *eligible*, if only eligible voters can contribute votes and furthermore each voter can cast at most a single vote.

Partial Universal Verifiability: We call an (M, m, t)-electronic voting scheme *partial universal verifiable* if every active or passive observer of the election is able to verify that only invalid votes are discarded.

Universal Verifiability: We call an (M, m, t)-electronic voting scheme *universal verifiable* if it is partial universal verifiable and furthermore each passive observer can check that the final result is consistent with the valid votes.

Robustness: We call an (M, m, t)-electronic voting scheme *robust* if every voter trying to cheat by handing in an invalid vote can be detected and discarded and any coalition of at most $t - 1$ tallying authorities cannot influence the result of the election.

Copy Protection: A voting scheme is *copy protected* if a voter cannot copy the vote of another voter.

Remark 2.1. In this work we do not consider receipt-freeness or non-coercibility, which was introduced in [BT94] and discussed in [CGS97].

2.1 Proofs of Knowledge and Partial Knowledge

In this section we generalize the notion of a Σ-protocol and customize the results of [CDS94] to obtain a protocol for a so-called OR-proof. The following definition is similar to the one given in [BCK+14], to the notion of a *gap Σ-protocol* in [AJL+12] and also captures the notions of protocols for the proof of knowledge and proof of correct multiplication given in [BDOZ11], even though it is not made explicit there. The main difference to a Σ-protocol is that we allow a soundness gap, that is a gap between the relation the prover has to know the witness for and the relation of which a verifier will be convinced after an accepting conversation. This necessity comes from the structure of the proofs of knowledge for cryptosystems based on learning with errors we use. Amortized there are better solutions which get along without a gap of this kind (see [DL12]), but in our setting each voter has to provide merely a single proof of validity and thus we can not make use of amortization techniques.

Definition 2.2 (Σ'-protocol). Let S_x, S_w, S_a, S_c be arbitrary sets, $R, R' \subseteq S_x \times S_w$ be binary relations such that $R \subseteq R'$ and Π be a 3-move protocol for R between a prover P and a verifier V of the following form:

$$\begin{tikzcd}
(x,w) \in R \arrow[d] & & x \in S_x \arrow[d] \\
\textbf{Prover } P & & \textbf{Verifier } V \\
\text{draw/calculate } a \in S_a \arrow[rrd, "a"] & & \\
& & \text{draw challenge } c \leftarrow S_c \arrow[lld, "c"'] \\
\text{calculate } e \in S_e \arrow[rrd, "e"] & & \\
& & \text{accept or reject}
\end{tikzcd}$$

Then Π is called a *Σ'-protocol for $R \subseteq R'$ with completeness error ε* if the following hold:

Completeness: If the prover P receives a tuple $(x, w) \in R$ as input, then the prover does abort with probability at most ε. The verifier accepts with overwhelming probability in case the prover does not abort.

Special Honest Verifier Computational Zero-Knowledge: For every $c \in S_c$ there exist values $a \in S_a$ and $e \in S_e$, such that the distribution of the conversation (a, c, e) is computationally indistinguishable to the conversation between an honest prover P and a verifier V drawing c as challenge. Furthermore these values can be generated efficiently.

Special Soundness: Let $(a, c, e), (a, c', e') \in S_a \times S_c \times S_e$ be any pair of conversations accepted by V with $c \neq c'$. Given those it is possible to efficiently extract a witness w for the public input x such that $(x, w) \in R'$.

Our goal is to transform an arbitrary Σ'-protocol into a Σ'-protocol, where the prover only proves partial knowledge. We will later us the special case of so called OR-proofs, where the prover proves knowledge of a witness for either a value v_1 or a value v_2. This enables a voter to prove that the ciphertext he provided either contains a 1 or a 0 and thus a valid yes- or no-vote.

First we need to introduce some notation. Let $R \subseteq S_x \times S_w$ be a binary relation and $k \in \mathbb{N}$. Then we define

$$R_k := \{((x_1, \dots, x_k), (w_1, \dots, w_k)) \in S_x^k \times S_w^k \mid \exists i \in [k] \colon (x_i, w_i) \in R\}.$$

The following theorem is a translation of the main result of [CDS94] to our setting.

Theorem 2.3. *Let Π be a Σ'-protocol for binary relations $R \subseteq R' \subseteq S_x \times S_w$ with completeness error ε and $k \in \mathbb{N}$. Furthermore let S be a smooth secret sharing scheme on S_c for k participants, such that the only qualified set is $[k]$ (compare Section 1.4). Then there exists a Σ'-protocol Π_k for the relations $R_k \subseteq R'_k$ with completeness error $1 - (1 - \varepsilon)^k$.*

Proof. Figure 2.2 describes the new protocol Π_k, where i denotes the index of the input of P such that $(x_i, w_i) \in R$. It is left to show that Π_k is indeed a Σ'-protocol.

Completeness can be seen straightforward. As the secret sharing scheme is assumed to be smooth, the challenge c_i is distributed uniformly random if the challenge c chosen by the verifier is distributed uniformly random. Therefore for index i an honest prover aborts with probability at most ε and for $j \neq i$ the simulator produces an aborting conversation with probability at most ε by the special honest verifier computational zero-knowledge property of Π. Therefore the prover aborts in total with probability at most $1 - (1 - \varepsilon)^k$ and otherwise produces accepting triples $(a_j, c_j, e_j)_j$.

To see *special honest verifier computational zero-knowledge* assume be given a value $c \in S_c$. Then the simulator proceeds as follows. For $j \in [k - 1]$ he draws $c_j \leftarrow S_c$ and calculates $c_k \in S_c$ such that the values $(c_j)_j$ form a sharing of c. Using the special honest verifier computation zero-knowledge property of Π he then calculates for all $j \in [k]$ the values a_j, e_j such that (a_j, c_j, e_j) is computationally indistinguishable to a conversation between an honest prover and the verifier executing Π. If one of the conversations simulates an abort, the simulator returns $\big((a_j)_j, c, \perp\big)$ and otherwise $\big((a_j)_j, c, ((c_j)_j, (e_j)_j)\big)$.

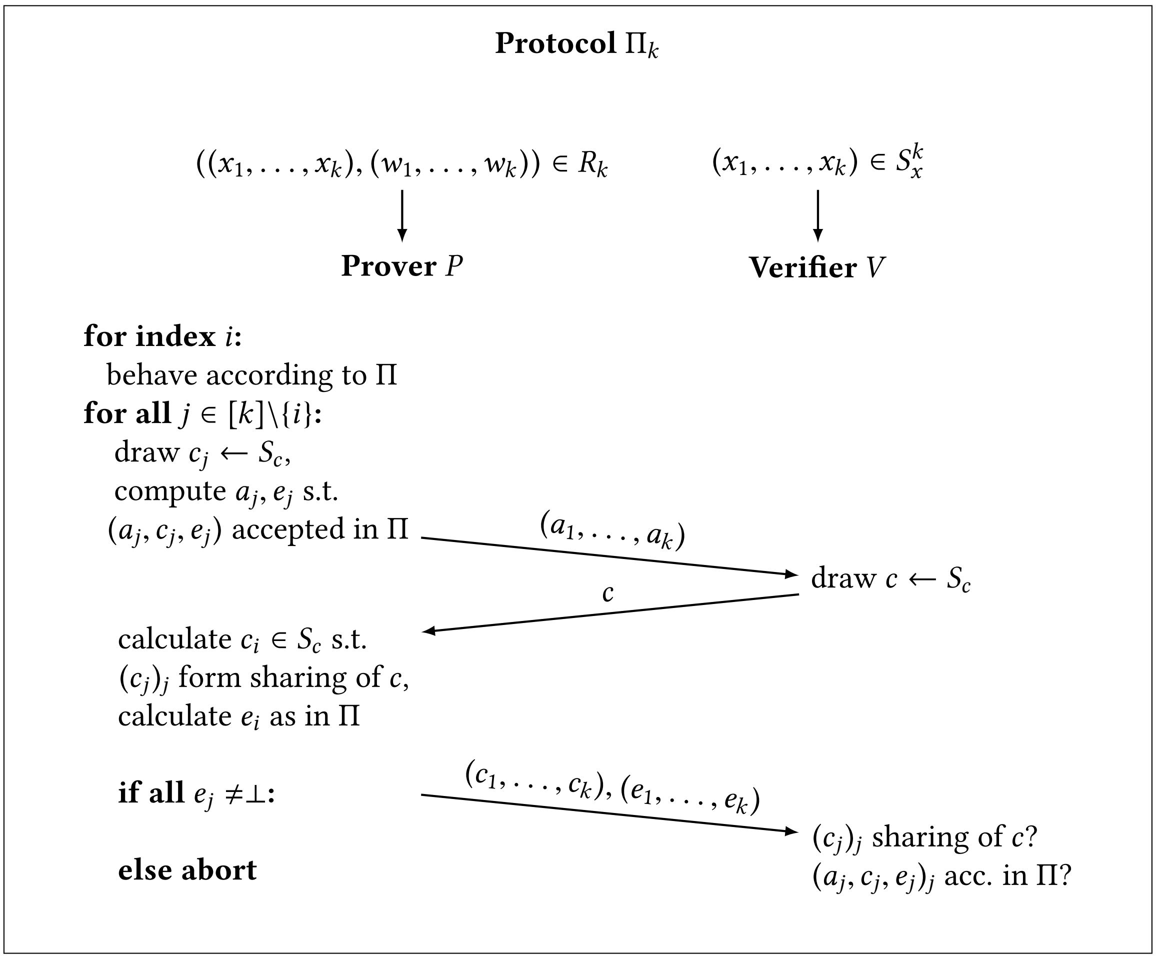
Protocol Π_k
$((x_1, \dots, x_k), (w_1, \dots, w_k)) \in R_k$
$(x_1, \dots, x_k) \in S_x^k$
Prover P
Verifier V
for index i**:**
behave according to Π
for all $j \in [k] \backslash \{i\}$**:**
draw $c_j \leftarrow S_c$,
compute a_j, e_j s.t.
(a_j, c_j, e_j) accepted in Π
$(a_1, \dots, a_k)$
draw $c \leftarrow S_c$
c
calculate $c_i \in S_c$ s.t.
$(c_j)_j$ form sharing of c,
calculate e_i as in Π
if all $e_j \neq \perp$**:**
$(c_1, \dots, c_k), (e_1, \dots, e_k)$
$(c_j)_j$ sharing of c?
$(a_j, c_j, e_j)_j$ acc. in Π?
else abort

Figure 2.2: Protocol for Proving Partial Knowledge (compare [CDS94], Theorem 8)

For $j \neq k$ the triples are exactly those produced during a real execution of Π_k, where the prover has knowledge of index w_k such that $(x_k, w_k) \in R$ and for the index k the computational indistinguishability follows from the computational indistinguishability of (a_k, c_k, e_k) to a triple generated during a real execution of Π.

For *special soundness* let $\big((a_j)_j, c, ((c_j)_j, (e_j)_j)\big)$ and $\big((a_j)_j, c', ((c'_j)_j, (e'_j)_j)\big)$ be two pairs of accepted conversations with $c \neq c'$. Since $[k]$ is a qualified set of the secret sharing scheme S, the values $(c_j)_j$ determine the value $c \in S_c$. Thus there must be an index $i \in [k]$ such that $c_i \neq c'_i$. Now we can use the special soundness property of Π and given the conversations (a_i, c_i, e_i) and (a_i, c'_i, e'_i) calculate a witness w_i such that $(x_i, w_i) \in R'$. Thus choosing arbitrary values $w_j \in S_w$ for all $j \in [k] \backslash \{i\}$ we have $((x_j)_j, (w_j)_j) \in R'_k$ by definition of R'_k. $\square$

2.2 Underlying Public Key Cryptosystem

As mentioned earlier the underlying public key cryptosystem has to comply with certain properties. For each $k \in \mathbb{N}$ let $B_r(k), B'_r(k), B(k) \in \mathbb{N}$ be natural numbers such that $B_r(k) \leq$

$B'_r(k)$. The following definition is inspired by the notion of a threshold homomorphic encryption scheme introduced in [CDN01].

Definition 2.4. We call PKE $:=$ (Gen, Enc, Dec) a *t-threshold B-additive homomorphic encryption scheme with (B_r, B'_r)-pre-proof of validity*, if the following holds.

Key Generation: Let $K := \{K_k^{pk} \times K_k^{sk_1} \times \cdots \times K_k^{sk_m}\}_{k \in \mathbb{N}}$ be a family of direct products of finite sets. Gen is a randomized algorithm, which takes 1^κ as input and returns a public key $pk \in K_\kappa^{pk}$ and a vector of secret key shares $(sk_1, \cdots, sk_m) \in K^{sk} := K_\kappa^{sk_1} \times \cdots \times K_\kappa^{sk_m}$. Further there exists an m-party protocol Π_{KeyGen} securely evaluating the key generator Gen assuming a static active adversary corrupting up to $t-1$ parties.

Encryption: Let $U := \{U_k\}_{k \in \mathbb{N}}$ be the space of randomness for encrypting and for each $k \in \mathbb{N}$ let $N_k \colon U_k \to \mathbb{R}$ a norm on U_k and D_k a distribution on U_k, such that for $r \leftarrow D_k$ it holds $N_k(r) \leq B_r(k)$ with overwhelming probability. Let $\mathcal{M} := \{\mathcal{M}_k\}_{k \in \mathbb{N}}$ the message space. The algorithm Enc takes as implicit input 1^κ a public key $pk \in K_k^{pk}$ and a message $m \in \mathcal{M}_\kappa$ as input and returns a ciphertext in C_κ, where $C := \{C_k\}_{k \in \mathbb{N}}$ is the space of ciphertexts.

Decryption: On implicit input 1^κ and explicit input $sk \in K_\kappa^{sk_1} \times \cdots \times K_\kappa^{sk_m}$ and $m \in C^\kappa$ the algorithm Dec returns a message $m \in \mathcal{M}_\kappa$ or $\perp$. Further there exits an m-party protocol Π_{Dec} securely evaluating the decryption algorithm Dec assuming a static active adversary corrupting up to $t-1$ parties.

Pre-Proof of Validity: For any $k \in \mathbb{N}$ there exists a constant $\varepsilon \in [0, 1/2)$ and a Σ'-Protocol $\Pi_{\mathrm{PPoV}}(k)$ for the relations

$$\begin{aligned} R_{\mathrm{PPoV}}(k) := \{(x, w) \mid & x = (pk, m, c) \in K_k^{pk} \times \mathcal{M}_k \times C_k \land w = r \in U_k \\ & \land N_k(r) \leq B_r(k) \land c = \mathrm{Enc}_{pk}(v; r)\} \end{aligned}$$

and

$$\begin{aligned} R'_{\mathrm{PPoV}}(k) := \{(x, w) \mid & x = (pk, v, c) \in K_k^{pk} \times \mathcal{M}_k \times C_k \land w = r \in U_k \\ & \land N_k(r) \leq B'_r(k) \land c = \mathrm{Enc}_{pk}(v; r)\} \end{aligned}$$

with knowledge error ε.

Additive Homomorphic Properties: For any $k \in \mathbb{N}$ given $B(k)$ ciphertexts $c_i := \mathrm{Enc}_{pk}(v_i; r_i)$ for messages $v_i \in \mathcal{M}_k$ and randomness r_i with $N_k(r_i) \leq B'_r(k)$ it holds

$$\mathrm{Dec}_{sk}\left(\sum_{i=1}^{B} c_i\right) = \sum_{i=1}^{B} v_i.$$

We will sometimes refer to his property by saying a public key encryption scheme is *roughly additive homomorphic.*

Threshold Semantic Security For each $k \in \mathbb{N}$, $I \subseteq [m]$ we define the following experiment.

```
**Experiment** $Exp^{t\text{−IND-CPA}}_{\text{PKE},\mathcal{A},I}$ :
$(pk, sk_1, \cdots, sk_m) \leftarrow \text{Gen}(1^k)$
$(m_0, m_1, state) \leftarrow \mathcal{A}(1^k, pk, I, (sk_i)_{i\in I})$
$b \leftarrow \{0, 1\}$
$c := \text{Enc}_{pk}(m_b)$
$b' \leftarrow \mathcal{A}(1^k, pk, I, (sk_i)_{i\in I}, c)$
**if** $b = b'$ **return** 1
**else return** 0
```

We require that for any probabilistic polynomial time adversary $\mathcal{A}$ returning always messages of the same size $|m_0| = |m_1|$ and for any set $I \subseteq [m]$ of size at most $t - 1$ there exists a negligible function negl: $\mathbb{N} \mapsto \mathbb{R}$ and a natural number $k_0 \in \mathbb{N}$ such that for any $k \geq k_0$ it holds

$$\Pr\left[Exp^{t\text{−IND-CPA}}_{\text{PKE},\mathcal{A},I} = 1\right] \leq \frac{1}{2} + \text{negl}(k).$$

To obtain a universal verifiable electronic voting scheme the multi-party computation protocol Π_{Dec} for joint decryption additionally has to comply with the following property.

Definition 2.5 (Auditable Correctness, [BDO14], Definition 1)**.** A multi-party computation protocol for evaluating a circuit C with $k \in \mathbb{N}$ inputs $x_1, \dots, x_k$ and possible output y satisfies *auditable correctness* if there exists an auditor $\mathcal{T}_{\text{audit}}$ which on input $\tau, C, x_1, \dots, x_k$ and y outputs *accept* y with overwhelming probability if τ is the transcript of an evaluation of C and furthermore $C(x_1, \dots, x_k) = y$, and *reject* otherwise.

Unfortunately the protocols we employ for decryption do not satisfy this property, thus in the following we will consider partial universal verifiability instead.

2.3 Multi-Authority Election Scheme

Let all used parameters be defined as in the previous sections. We can now describe the (M, m, t)-multi-authority election scheme following the approach of [CGS97] as a framework to be instantiated with a suitable cryptosystem. Let PKE := (Gen, Enc, Dec) be a t-threshold, B-additive homomorphic encryption scheme with (B_r, B'_r)-pre-proof of validity Π_{PPoV} with negligible soundness error. Let Π_{PoV} be the Σ'-protocol obtained by applying Theorem 2.3 to Π_{PPoV} with 2 participants. To make the casting of votes non-interactive we assume the existence of a trusted *beacon*, that posts random bits over time and take the challenge from those bits. The beacon can be implemented by a hash function which takes a voter-specific input comprising the first message of the three-move proof; in this case security in the random oracle model is retained. For the second phase we assume

$M < B$. If this is not the case the votes are split up in groups of each B votes and the tallying authorities will first compute intermediary results, which are then added up to the final outcome. An (M, m, t)-electronic voting scheme is obtained by the following procedure, where $v_i \in \{0, 1\}$ denotes the vote of voter V_i for each $i \in [M]$. Before the election the m tallying authorities jointly execute the protocol Π_{KeyGen}, save their respective private share of the secret key and broadcast the public key.

Phase I: Casting Votes

1. For each $i \in [M]$ voter V_i encrypts his vote $v_i \in \{0, 1\}$ by encrypting it to $c_i := \text{Enc}_{pk}(v_i)$ and executes $\Pi_{\text{PoV}}(pk, v_i, c_i)$ with randomness provided by the beacon.
2. Each voter posts his vote c_i accompanied by the proof of validity on his respective section of the bulletin board.

Phase II: Tallying

1. After the deadline all proofs of validity are checked and non-valid votes are discarded. The valid votes are summed up to $c := \sum_{i \in I} c_i$, where $I \subseteq [M]$ denotes the set of indices corresponding to valid votes.
2. The m tallying authorities jointly execute Π_{Dec} with input c and their respective key shares to obtain final result.

Theorem 2.6 (Compare [CGS97], Theorem 1)**.** *If the underlying cryptosystem* PKE *is a t-threshold, B-additive homomorphic encryption scheme with (B_r, B'_r)-pre-proof of validity with negligible soundness error, then the described (M, m, t)-electronic voting scheme provides privacy, partial universal verifiability, robustness and copy protection.*

Proof. To prove privacy of the voting scheme we first consider the published ciphertext and then the accompanied proof of validity.

Let $\mathcal{B}$ a coalition of at most $t - 1$ collaborating parties such that the probability $\Pr[\mathcal{B}(1^k, pk, c) = v \mid c = \text{Enc}_{pk}(v) \wedge v \in \{0, 1\}]$ is noticeable better than guessing. Let $I_C \subseteq [m]$ be the set of indices of the tallying authorities within the collaborating parties. Then from $\mathcal{B}$ we can construct an adversary $\mathcal{A}$ winning $Exp^{t\text{-IND-CPA}}_{\text{PKE},\mathcal{A},I_C}$ (Definition 2.4) with noticeable probability better than guessing as follows. The adversary $\mathcal{A}$ returns the messages $v_0 = 0$ and $v_1 = 1$. Finally $\mathcal{A}$ starts $\mathcal{B}$ with the challenge ciphertext as input and returns the output of $\mathcal{B}$. As $\mathcal{B}$ has access to at most $t-1$ secret key shares, this contradicts the t-threshold semantic security of the underlying cryptosystem.

It is left to consider the proof of validity Π_{PoV} obtained from Π_{PPoV} as explained in the introduction of this section. Since the protocol Π_{PoV} is honest verifier zero-knowledge, it is also witness indistinguishable. This follows from the fact that the distribution of a conversation using an arbitrary witness is indistinguishable to the distribution of a conversation produced by a simulator without knowledge of any witness. Thus the proof of validity does not leak information about the encrypted vote either and altogether the voting scheme provides privacy.

Partial universal verifiability is achieved as the non-interactive proofs posted by each voter can be checked by any observer of the election.

It remains to prove robustness. Each invalid vote can be detected and discarded with overwhelming probability, because the soundness-error of the proof accompanying each vote is negligible. As further by assumption the protocols Π_{KeyGen} and Π_{Dec} are secure against an active adversary corrupting up to $t-1$ parties, a coalition of parties not exceeding this size cannot influence the result of the election.

□

3 Public Key Encryption Based on RLWE

In this chapter we give an overview on how to implement the electronic voting scheme with cryptosystems based on RLWE. The crucial point is to achieve active security during distributed decryption. Our approach is to start with a decryption protocol which complies with a certain structure and is secure against passive adversaries. We show how to achieve security against active adversaries by employing the protocols for secure multi-party computation presented in [BDOZ11] and [DPSZ12] in Chapter 4. We first give a general framework for cryptosystems based on RLWE and then present NTRU as the instantiation of our choice. The obtained electronic voting scheme is efficient, eligible, robust, partial universal verifiable and further provides copy protection and privacy based on learning with errors over rings.

3.1 Abstract Description of the Cryptosystem

In this section we give the framework for an abstract encryption scheme complying with a weakened version of the requirements stated in Definition 2.4. It is fitted to be instantiated with a cryptosystem based on learning with errors over rings. Together with the results of Chapter 4 we will thereby obtain a cryptosystem for instantiating the electronic voting scheme presented in Section 2.3.

Let $m := m(\kappa) \in \mathbb{N}$ be the number of authorities and let $t := t(\kappa) \in \mathbb{N}$ such that at most $t - 1$ authorities are under control of a static adversary. Note that depending on the multi-party computation of chapter 4 to be used $t - 1 < m/2$ may be required.

Let $B(\kappa), B_r(\kappa), B'_r(\kappa) \in \mathbb{N}$ such that $B_r \leq B'_r$. Let PKE be a t-threshold, B-additive homomorphic encryption scheme with (B_r, B'_r)-pre-proof of validity with the weakening that Π_{Dec} is merely required to implement the functionality $\mathcal{F}_{\mathrm{PDec}}$ (Figure 3.2) in the $\mathcal{F}_{\mathrm{KeyGen}}$-hybrid model (Figure 3.1) against a static active adversary corrupting up to $t - 1$ parties.

The functionality $\mathcal{F}_{\mathrm{PDec}}$ allows the adversary to freely choose the result of the decryption, intuitively this corresponds to only requiring security against passive adversaries. A *passive* adversary obtains all messages and internal data of corrupted parties, but apart from that corrupted parties behave according to the protocol. In other words Π_{Dec} is only required to be secure against the leakage of information but not giving any guarantees on the correctness of decryption.

The functionality $\mathcal{F}_{\mathrm{KeyGen}}$ captures that we consider threshold cryptosystems where the key is either Shamir shared or additively shared between the authorities. In the latter case the secret keys has to be shared using polynomials of different degrees. This will be crucial to prevent the leakage of information about the secret key by adding smudging values (see Theorem 3.6). Note that if an public key encryption scheme is IND-CPA secure

Functionality $\mathcal{F}_{\text{KeyGen}}$

- Upon receiving "*start*" from all honest players, run the algorithm for key generation $(pk, sk) \leftarrow \text{Gen}(1^\kappa)$ and send pk to the adversary.
- *Additive Sharing:* Receive a share $sk_i \in S_{sk}$ for each $i \in I_C$, that is for each corrupt player P_j. Choose for each honest player P_i a share $sk_i \in K_\kappa^{sk_i}$ such that $(sk_i)_{i\in[m]}$ forms an additive sharing of the secret key sk, that is $sk = \sum_{i=1}^m sk_i$. Send the public key pk to each player and additionally to each honest player P_i the share sk_i.
- *Shamir Sharing:* For each $j \in \{t, \dots, m\}$ receive a share $sk_i^j \in K_\kappa^{sk_i}$ for each $i \in I_C$ from the adversary. Calculate for each honest player P_i shares $sk_i^j \in K_\kappa^{sk_i}$ as in Remark 1.25 such that $\left(sk_i^j\right)_{i\in[m]}$ forms a (j, m)-Shamir sharing of the secret key. Send the public key pk to each player and additionally to each honest player P_i the set of shares $\left(sk_i^j\right)_{j\in\{t,\dots,m\}}$.

Figure 3.1: Ideal Functionality for Distributed Key Generation (compare [DPSZ12], Figure 2 and [BD10], Section 4.1)

the threshold semantic security follows directly from the security of the secret sharing scheme.

If the cryptosystem PKE now additionally complies with the properties stated in the following, the correctness of decryption can be ensured with the protocols established in Chapter 4.

Let $p := p(\kappa) \in \mathbb{P}$ be a prime, $N := N(\kappa) \in \mathbb{N}$ a natural number and $\mathcal{M} \subseteq \mathbb{F}_p[X]/\phi_N$ the message space, where ϕ_N is the N-th cyclotomic polynomial. Let $P := P(\kappa) \in \mathbb{N}$ be a prime power, $s_1 := s_1(\kappa) \in \mathbb{N}$ a natural number and the ciphertext space of the form $C := (\mathbb{F}_P[X]/\phi_N)^{s_1}$. The requirement on P to be a prime power is necessary, as the protocols presented in [BDOZ11] and [DPSZ12] only support computation of arithmetic circuits over finite fields. Note that we require P to be super-polynomial, as we need to statistically hide error terms to not leak information about the respective secret key shares during decryption. As explained in the introduction this is the main reason why we cannot directly base the security of the election scheme in this setting on worst-case lattice problems. For the multi-party setting let $m := m(\kappa) \in \mathbb{N}$ be the number of parties and $t := t(\kappa) \in \mathbb{N}$ such that at most $t - 1$ players are corrupted by the static adversary. We will denote the index set of honest players by $I_H \subseteq [m]$ and the index set of corrupted players by $I_C \subseteq [m]$.

We require that Π_{Dec} is of the following form. First each player P_i locally computes a publicly known arithmetic circuit C with output in $\mathbb{F}_P/[X]\phi_N$, where the input consists of the shares of the secret key sk owned by P_i and the publicly known ciphertext c. Then

Functionality $\mathcal{F}_{\text{PDec}}$

- Upon receiving "*start*" from all honest players, run the algorithm for key generation $(pk, sk) \leftarrow \text{Gen}(1^{\kappa})$, send pk to the adversary and store sk.
- Upon receiving "*decrypt c*" from all honest players (or in the Shamir setting "*decrypt c, I*" from a set $I \subseteq [m]$ of at least t players), send c and $v := \text{Dec}_{sk}(c) \in \mathcal{M} \cup \{\perp\}$ to the adversary.
- On receiving $v' \in \mathcal{M} \cup \{\perp\}$ from the adversary, send "*result v'*" to all players.

Figure 3.2: Ideal Functionality for Distributed Decryption (see [DPSZ12], Figure 3)

each player adds a bounded value to his share and sends the result to the other players, which allows the players to jointly recover the plaintext encrypted in c.

For the use of the threshold variant of the protocol of Bendlin et al. (Section 4.1) there are additional requirements on Π_{Dec}. Let I be the set of indexes of players taking part in the decryption process, for each $i \in I$ let $d_i := d_{i,I}$ be defined as in Remark 1.24 and further let $B \in \mathbb{N}$ be a natural number and

$$I_B := \{\rho \in \mathbb{F}_P[X]/\phi_N \mid \|\rho\| \leq B\}$$

the set of all polynomials with norm bounded by B. Then Π_{Dec} has to comply with the following form.

- For each $i \in I$ the player P_i calculates $r_i := C(sk_i^{|I|}, c) \in \mathbb{F}_P[X]/\phi_N$. The operations of the circuit C are restricted to additions of shares and additions and multiplication of publicly known constants to shares.
- For every $i \in I$ player P_i finally chooses a value $a_i \in I_B$ uniformly random, calculates $r_i' := r_i + d_i \cdot a_i \in \mathbb{F}_P[X]/\phi_N$ and broadcasts it. The plaintext can then be publicly recovered from the broadcasted values.

3.2 NTRU Encryption Scheme

In this section we present the so-called NTRU, which was originally introduced in [HPS98]. It was modified in [SS11] to achieve security under the assumed hardness of standard lattice-based problems. In our setting this security reduction is not directly applicable, as it requires the modulus to be polynomially bounded. Instead we use those results to base the security and correctness of NTRU on the learning with errors over rings assumption.

We begin by presenting the definition of NTRU based on [SS11] and [BCK+14] and proceed giving the intuition for security and additive homomorphic properties from [SS11].

Finally we provide a protocol for passive secure decryption of the required form based on [BD10] and [DPSZ12] and establish the pre-proof of validity with the methods of [BCK+14].

Let $N := N(\kappa) \in \mathbb{N}$ be of the same order of magnitude as κ. The ring we will be working with is of the form $\mathbb{Z}[X]/\phi_N$, where $\phi_N \in \mathbb{Z}[X]$ is the N-th cyclotomic polynomial. For our purposes we demand that N is a power of two. Recall that in this case we have $\phi_N = X^n + 1$ for $n = N/2$. Let $P := P(\kappa) \in \mathbb{P}$ super-polynomial and define $R := \mathbb{Z}[X]/(X^n + 1)$ and $R_P := \mathbb{Z}_P[X]/(X^n + 1)$.

Definition 3.1 (NTRU Scheme). Let $p := p(\kappa) \in \mathbb{P}$ be a prime such that $p < P$ and $\mathcal{M} := \{v \in R \mid \|v\|_\infty < p\}$. Let further $\sigma := \sigma(\kappa), \alpha := \alpha(z\kappa) \in \mathbb{R}$.

The public key encryption scheme NTRU consists of three polynomial time algorithms Gen, Enc and Dec complying with the following.

Key Generation: The algorithm Gen draws elements s' and g according to the discrete Gaussian distribution D_σ on R_P and sets $s := p \cdot s' + 1$. It checks whether s and g are invertible in R_P and repeats if not. When eventually $s, g \in R_P^\times$ the algorithm outputs the secret key

$$s \in R_P^\times$$

and the public key

$$p \cdot g \cdot s^{-1} \in R_P.$$

In the following we will work with the ideal functionality $\mathcal{F}_{\text{KeyGen}}$ (Figure 3.1) for distributed key generation. An implementation can be obtained using generic techniques of multi-party computation.

Encryption: The algorithm Enc takes a public key $h \in R_p$ and a message $r \in \mathcal{M}$ as input, draws e and f according to D_α and returns the ciphertext

$$h \cdot e + p \cdot f + v \in R_P.$$

Decryption: Given a secret key $s \in R_p$ and a ciphertext $y \in R_P$ as input the decryption algorithm Dec returns

$$s \cdot y \bmod p.$$

3.2.1 Security and Proof of Validity

The goal of this section is to outline the results of [SS11] on how to choose the parameters of NTRU such that the security can be reduced to the learning with errors over rings assumption. Further we use the results of [BCK+14] to establish a pre-proof of validity which can be turned into a proof of validity as explained in Section 2.3. The correctness of the NTRU encryption scheme will follow from the additive homomorphic properties we prove in Lemma 3.5.

Let in the following all parameters not explicitly mentioned be defined as in the introduction of section 3.2 and Definition 3.1. The first lemma outlines under which circumstances the distribution of the public key is statistical indistinguishable from uniformly random distribution. For the proof we refer to [SS11].

Lemma 3.2 (Public Key Uniformity ([SS11], Theorem 3)). *Let $\varepsilon \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{R}$ such that $\sigma \geq 2n\sqrt{\ln(8nP)} \cdot P^{1/2+2\varepsilon}$. Then the public key generated according to the key generation algorithm of the version of* NTRU *presented in 3.1 is distributed within statistical distance of $2^{3n}P^{-\lfloor \epsilon n \rfloor}$ to uniformly random.*

The following lemma of [SS11] gives a reduction from the RLWE decision problem to the security of the NTRU encryption scheme.

Lemma 3.3 (IND-CPA security (compare [SS11], Lemma 13)). *Let $\varepsilon \in \mathbb{R}_{>0}$ and $\sigma \in \mathbb{R}$ such that $\sigma \geq 2n\sqrt{\ln(8nP)} \cdot P^{1/2+2\varepsilon}$. If there exists a $\delta \in \mathbb{R}_{>0}$ and an IND-CPA attack against the* NTRU *version presented in 3.1 that runs in time T and has success probability $1/2 + \delta$, then there exists an algorithm deciding $RLWE^{\times}_{\mathrm{HNF}}(\phi, \alpha, P)$ in time $T' \in O(T + n)$ and with success probability statistically close to δ.*

Proof. Assume $\mathcal{A}_{\text{IND-CPA}}$ is an IND-CPA attack algorithm running in time T with success probability $1/2+\delta$. Then we construct an algorithm $\mathcal{A}_{\text{RLWE}}$ that decides $\text{RLWE}^{\times}_{\text{HNF}}(\phi, \alpha, P)$ as follows. Let $(a, x) \in R_P^{\times} \times R_P$ be a sample given to $\mathcal{A}_{\text{RLWE}}$. Now $\mathcal{A}_{\text{RLWE}}$ starts $\mathcal{A}_{\text{IND-CPA}}$ with public key $h := p \cdot a \in R_P$ as input. Let $v_0, v_1 \in \mathcal{M}$ be the challenge messages outputted by $\mathcal{A}_{\text{IND-CPA}}$. Then $\mathcal{A}_{\text{RLWE}}$ draws a random bit $b \leftarrow \{0, 1\}$ and returns $y := p \cdot x + v_b$ to $\mathcal{A}_{\text{IND-CPA}}$. Finally $\mathcal{A}_{\text{RLWE}}$ outputs 1 if the algorithm $\mathcal{A}_{\text{IND-CPA}}$ guesses the correct bit, implying the sample was chosen according to $A_{e,\alpha}$. Otherwise $\mathcal{A}_{\text{RLWE}}$ outputs 0, implying the sample was chosen uniformly random.

As p is invertible in R_P and a chosen uniformly random from $R_P^{\times}$, by Lemma 3.2 the public key given as input to $\mathcal{A}_{\text{IND-CPA}}$ is statistically indistinguishable from an honestly generated key. If the sample was chosen according to $A_{e,\alpha}$ the second value x is of the form $x = ae + f$ for values e, f chosen according to the distribution D_α and thus y has the same distribution as an honestly generated encryption of v_b, thus $\mathcal{A}_{\text{IND-CPA}}$ succeeds and $\mathcal{A}_{\text{RLWE}}$ returns 1 with probability statistically close to $1/2 + \delta$.

In case the sample was chosen uniformly random, the value y is uniformly random and independent of b and thus in this case $\mathcal{A}_{\text{RLWE}}$ returns 1 with probability $1/2$. Together this proves the claim.

□

Now we present a Σ'-protocol for the pre-proof of validity as required. We will therefore customize the proof of knowledge given in [BCK+14]. The proof is more efficient compared to the pre-proof of validity for cryptosystems based on learning with errors we present in Chapter 5 as the challenge space is larger. Furthermore the proof is based on rejection sampling instead of smudging, thereby avoiding a soundness gap of super-polynomial size. Let $\tau = \tau(n) \in \omega(\log n)$ and $\lambda = \lambda(n) \in \omega(\tau\alpha\sqrt{\log(n)})$. Formally we will present a Σ'-protocol for the relations

$$\begin{aligned} R_{\text{NTRU-PPoV}} := \{(x, w) \mid & x = (h, p, y, v) \in R_P \times \mathbb{P} \times R_P \times \mathcal{M} \wedge w = (e, f) \in R_P \times R_P \wedge \\ & y = he + pf + v \wedge \|e\|_\infty, \|f\|_\infty \leq \tau\alpha\} \end{aligned}$$

and

$$\begin{aligned} R'_{\text{NTRU-PPoV}} := \{(x, w) \mid & x = (h, p, y, v) \in R_P \times \mathbb{P} \times R_P \times \mathcal{M} \wedge w = (e, f) \in R_P \times R_P \wedge \\ & 2y = he + pf + 2v \wedge \|e\|_\infty, \|f\|_\infty \leq 2\tau\lambda n\}. \end{aligned}$$

Protocol $\Pi_{\text{NTRU-PPoV}}$

$$\begin{tikzcd}[column sep=huge]
x = (h,p,y,v),\ w = (e,f) \arrow[d] & & x = (h,p,y,v) \arrow[d] \\
\textbf{Prover } P & & \textbf{Verifier } V \\
\begin{array}{l} r_e, r_f \leftarrow D_\lambda \\ a := hr_e + pr_f \\ (\gamma,\delta) := \mathrm{Com}(a) \end{array} \arrow[rr, "\gamma"] & & c \leftarrow \{0,\dots,2n-1\} \arrow[dll, "c"'] \\
\begin{array}{l} \varepsilon := r_e + eX^c \in R_P \\ \zeta := r_f + fX^c \in R_P \end{array} & & \\
\textbf{with probability } \rho: \arrow[rr, "{(a,\delta,\varepsilon,\zeta)}"] & & \begin{array}{l} \mathrm{open}(\mathrm{Com}(a,\gamma,\delta)) \overset{?}{=} 1 \\ a + yX^c \overset{?}{=} h\varepsilon + p\zeta + vX^c \\ \|\varepsilon\|_\infty, \|\zeta\|_\infty \overset{?}{\leq} \tau\lambda \end{array} \\
\textbf{else } \text{abort} & &
\end{tikzcd}$$

with $\rho := \frac{D_\lambda(\varepsilon)\cdot D_\lambda(\zeta)}{M^2\cdot D_{eX^c,\lambda}(\varepsilon)\cdot D_{fX^c,\lambda}(\zeta)}$

Figure 3.3: Pre-Protocol for Proving Validity of a Vote (compare [BCK+14], Protocol 3.2)

Note that the relations deviate from the requirements on the abstract cryptosystem established in Section 2.2. This issue can be solved by using twice the ciphertexts and dividing the final result after decryption by two.

Lemma 3.4. *Let $M \in O(1)$ be as in Theorem 1.9 with $T = \tau\alpha$, $\sigma = \lambda \in \omega(T\sqrt{\log(n)})$ and the discrete Gaussian distribution $D = D_\alpha$ on R_P. Let* Com *be a commitment scheme, which is perfectly binding and computationally hiding. Then under the respective assumption the protocol Π_{PPoV} given in Figure 3.3 is a Σ'-protocol for the relations $R_{\text{PPoV}} \subseteq R'_{\text{PPoV}}$ with completeness error* $1 - 1/M^2$.

Proof. Let all parameters be defined as in Figure 3.3. We show some of the required equalities and inequalities only for the parameters e, ε and r_e, because the proofs for f, ζ and r_f work completely analogous.

Completeness: By our choice of parameters and Theorem 1.9 every honest prover responds to the verifier with probability statistically close to $1 - 1/M^2$. Furthermore in this case the equations to be checked by the verifier hold. Let P be an honest prover having knowledge of randomness e, f for the message v. Then we have

$$a + yX^c = hr_e + pr_f + heX^c + pfX^c + vX^c = h\varepsilon + p\zeta + vX^c$$

and the inequalities hold since

$$\|\varepsilon\|_\infty \leq \|r_e\|_\infty + \|e\|_\infty \leq \frac{1}{2}\tau\lambda + \frac{1}{2}\tau\alpha$$

and

$$\|\varepsilon\|_\infty \leq \tau\lambda$$

with overwhelming probability by Lemma 1.5.

Special honest-verifier computational zero-knowledge: Let $c \in \{0, \ldots, 2n-1\}$ be a challenge given to the simulator. The simulator first draws $\varepsilon, \zeta \leftarrow D_\lambda$, calculates $a := h\varepsilon + p\zeta - yX^c + vX^c$ and sets $(\gamma, \delta) := \mathrm{Com}(a)$. Then he returns

$$(\gamma, c, (a, \delta, \varepsilon, \zeta))$$

with probability $1/M^2$. By Theorem 1.9 outputting the obtained values ε, ζ with probability $1/M^2$ is statistically indistinguishable from outputting the honestly generated values with probability ρ. Note that the quadratic factor comes from applying rejection sampling twice. As the value a of an accepting conversation can be derived deterministically from those values it is indistinguishable to the value generated during a real execution. Thus the distribution of the transcript is indistinguishable to a real protocol transcript where no abort occurs. With probability $1 - 1/M^2$ the simulator outputs

$$(\gamma, c, \perp)$$

simulating an abort. As we assume Com to be computationally hiding, the distribution of this protocol is computationally indistinguishable to a real protocol transcript where an abort occurs. We have $D_{eX^c,\lambda}(r_e + eX^e) = D_\lambda(r_e)$ and furthermore for $r_e \leftarrow D_\lambda$ we have $D_\lambda(r_e + eX^e) = D_\lambda(r_e + e)$ as the components of r_e are distributed identically. Thus the probability that during an honest execution an abort occurs is independent of c. Since furthermore the probability that during an honest execution an abort occurs is statistically close to $1 - 1/M^2$ the output of the simulator is statistically indistinguishable to the conversation between an honest prover and a verifier.

Special soundness: Let $(\gamma, c, (a, \delta, \varepsilon, \zeta))$ and $(\gamma, c', (a', \delta', \varepsilon', \zeta'))$ be two accepting conversations with $c \neq c'$. By the binding property of the commitment scheme we have $a = a'$. Further the second verification equation yields

$$(y - v)(X^c - X^{c'}) = h(\varepsilon - \varepsilon') + p(\zeta - \zeta')$$

and thus with $\tilde{e} := 2(X^c - X^{c'})^{-1}(\varepsilon - \varepsilon')$ and $\tilde{f} := 2(X^c - X^{c'})^{-1}(\zeta - \zeta')$ we have

$$2y = h\tilde{e} + p\tilde{f} + 2v.$$

Finally by Lemma 1.19 and 1.20 we find

$$\|\tilde{e}\|_\infty = \|2(X^c - X^{c'})^{-1}(\varepsilon - \varepsilon')\|_\infty \leq n\|\varepsilon - \varepsilon'\|_\infty \leq n2\tau\lambda.$$

□

3.2.2 Distributed Decryption

We start this section by showing that NTRU is roughly additive homomorphic for a suitable choice of parameters. That followed we provide a protocol for distributed decryption complying with the properties established in Section 3.1. Our approach is based on the protocols for distributed decryption presented in [BD10] and [DPSZ12].

Let $\tau \in \omega(\log n)$ arbitrary and $B'_\sigma := \tau\sigma$. Then for a honestly generated public key $h = p \cdot g \cdot s^{-1} \in R_P$ we have $\|g\|_\infty \leq B'_\sigma$ and $\|s\|_\infty \leq p \cdot B'_\sigma + 1$ with overwhelming probability by Lemma 1.5. Let further $B'_r := 2\tau^{2.5}\alpha n$ be the bound of randomness an honest prover can convince the verifier of with the proof of knowledge established in previous section. Let $L' := p \cdot n \cdot (B'_\sigma \cdot B'_\alpha(p+1) + p \cdot B'_\sigma + B'_\alpha + 1)$. Then we can prove the following Lemma.

Lemma 3.5 (Additive Homomorphic Properties). *Let $B'_\sigma \in \mathbb{N}$, $B'_r \in \mathbb{N}$ and $L' \in \mathbb{N}$ as above. Let $B \in \mathbb{N}$, $L := B \cdot L'$ and for $i \in [B]$ let $y_i := \mathrm{Enc}_h(v_i; e_i, f_i)$ be ciphertexts with $\|e_i\|, \|f_i\| < B'_r$, $\|v_i\|_\infty \leq p$ for all $i \in [B]$. In case*

$$L + 2^u \cdot L < P/2$$

for any $x \in R_p$ with $\|x\|_\infty \leq 2^u \cdot L/p$ we find

$$\mathrm{Dec}_s\left(\sum_{i=1}^{B} y_i\right) = \sum_{i=1}^{B} v_i$$

even if $p \cdot x$ is added before calculating modulo p. Note that the additional value x will correspond to the smudging error added during distributed decryption for preventing leakage of information about the secret key.

Proof. In the ring R_P we have

$$\begin{aligned} s \cdot \sum_{i=1}^{B} y_i &= s \cdot \left(h \cdot \sum_{i=1}^{B} e_i + p \cdot \sum_{i=1}^{B} f_i + \sum_{i=1}^{B} v_i\right) \\ &= \underbrace{p \cdot \left(g \cdot \sum_{i=1}^{B} e_i + s \cdot \sum_{i=1}^{B} f_i + s' \cdot \sum_{i=1}^{B} v_i\right) + \sum_{i=1}^{B} v_i}_{=:\zeta} . \end{aligned}$$

By ζ we now consider the term in R without reducing modulo P. Decryption yields the correct result if it holds $\|\zeta\|_\infty < P/2 - 2^u L$, as then after reduction modulo p only the sum of the plaintext messages remains and furthermore adding $p \cdot x$ does not lead to an overflow. With Lemma 1.20 we obtain

$$\begin{aligned} \|\zeta\|_\infty &\leq p \cdot \left(n \cdot \|g\|_\infty \cdot \sum_{i=1}^{B} \|e_i\|_\infty + n \cdot \|s\|_\infty \cdot \sum_{i=1}^{B} \|f_i\|_\infty\right) + n \cdot \|s\|_\infty \cdot \sum_{i=1}^{B} \|v_i\|_\infty \\ &\leq p \cdot n \cdot B \cdot (B'_\sigma \cdot B'_r(p+1) + p \cdot B'_\sigma + B'_r + 1) < P/2 - 2^u \cdot L. \end{aligned}$$

□

Protocol $\Pi_{\text{NTRU-Dec}}$

Initialization: For each $i \in I$ the player P_i calculates

$$a_i := s_i^{|I|} \cdot y \in R_P,$$

where $s_i^{|I|} \in R_P$ is the respective share of player P_i of the secret key and for additive sharing $s_i^{|I|} := s_i$ respectively. Then P_i calculates $b_i := a_i + p \cdot d_i \cdot r_i$, where each coefficient of the polynom $r_i \in R$ is chosen uniformly random from the discrete interval $[-2^u \cdot L/(|I| \cdot p), 2^u \cdot L/(|I| \cdot p)]$.

Decryption: For each $i \in I$ the player P_i broadcasts b_i and every player computes the encrypted message by recovering the additively or $(|I|, m)$-Shamir shared value b and then calculating $b \bmod p$.

Figure 3.4: Protocol for Distributed Decryption of NTRU Ciphertexts (compare [DPSZ12] (Full Version), Appendix A.6 and [BD10], Section 4.2)

The protocol for distributed decryption is based on the techniques of [BD10] and [DPSZ12] and can be found in Figure 3.4 with $I := [m]$ and $d_i := 1$ for all $i \in [m]$ or alternatively with $I \subseteq [m]$ the set of players wanting to decrypt of size at least t and $d_i := d_{i,I} \in \mathbb{Z}_P$ as in Remark 1.24 for all $i \in I$.

Theorem 3.6 (Compare [DPSZ12] (Full Version), Appendix A.6 and [BD10], Theorem 3)**.** *The protocol* $\Pi_{\text{NTRU-Dec}}$ *(Figure 3.4) implements* $\mathcal{F}_{\text{PDec}}$ *(Figure 3.2) in the* $\mathcal{F}_{\text{KeyGen}}$*-hybrid model with statistical security against any static active adversary corrupting up to* $t-1$ *parties if* $L + 2^u \cdot L < P/2$ *for* L *as in Lemma 3.5.*

Proof. The requirement $L + 2^u \cdot L < P/2$ makes sure that the correct result is returned if all players behave according to the protocol by Lemma 3.5. The simulator proceeds as follows.

Simulator $\mathcal{S}_{\text{NTRU-Dec}}$**:**

Key Generation: When receiving "*start*" the simulator $\mathcal{S}$ forwards the command to the ideal functionality $\mathcal{F}_{\text{PDec}}$ and obtains the public key pk.
Additive Sharing: The simulator obtains for each $i \in I_C$ a share $sk_i \in R_P$, where I_C denotes the set of indices of at most $t-1$ players corrupted by the adversary and I_H the set of honest players. After receiving those values the simulator chooses for all $i \in I_H$ shares $sk_i \in R_P$ such that $\sum_{i=1}^m sk_i = 0$. For uniform notation set $sk_i^m := sk_i$ for all $i \in [m]$.
Shamir Sharing: The simulator obtains for each $i \in I_C$ a set of values $(sk_i^j)_{j\in\{t,\dots,m\}}$ from the adversary. After receiving those values the simulator chooses for all $i \in I_H$

set of shares $(sk_i^j)_{j\in\{t,\dots,m\}}$ such that for each $j \in \{t, \dots, m\}$ the values $(sk_i^j)_{i\in[m]}$ form an (j, m)-Shamir sharing of $0 \in R_P$.

Decryption: When receiving "*decrypt c*" or "*decrypt c, I*" the simulator $\mathcal{S}$ forwards the command to the functionality $\mathcal{F}_{\text{PDec}}$ and obtains $v = \text{Dec}_{sk}(c)$. Let $i^\star \in I_H$ be the index of an arbitrary honest player in I. Then the simulator computes for all (corrupted and honest) players P_i with index $i \in I\backslash\{i^\star\}$ the value a_i according to $\Pi_{\text{NTRU-Dec}}$ with help of the shares $(sk_i^{|I|})_{i\in I\backslash\{i^\star\}}$ of the secret key received or simulated during key generation. He determines $a_{i^\star}$ such that the values $(a_i)_{i\in I}$ form an additive respectively $(|I|, m)$-Shamir sharing of v. For all $i \in I_H$ the simulator can now calculate b_i according to the protocol $\Pi_{\text{NTRU-Dec}}$ and finally broadcast the values $(b_i)_{i\in I_H}$. He receives $(b'_i)_{i\in I_C}$ from the adversary, calculates the shared value b' and sends $m' := b' \mod p$ to the functionality $\mathcal{F}_{\text{PDec}}$, which in turn sends *result m'* to all players and the adversary.

By the smoothness of the secret sharing schemes, the shares of the $|I| - 1$ players with indices in $I\backslash\{i^\star\}$ of $0 \in R_P$ are indistinguishable to shares of the secret key and therefore the values $(b_i)_{i\in I\backslash\{i^\star\}}$ generated by the simulator according to the protocol with the simulated shares are indistinguishable to the values calculated during an execution of the protocol. It remains to show that this also holds for the value $b_{i^\star}$. As this values can be derived deterministically from the other shares together with the value b, it is sufficient to show that the value b calculated by the simulator is indistinguishable from the value produced during a real execution of the protocol. This holds as $b = v + p \cdot \sum_{i\in I} r_i$ is statistically indistinguishable to $s \cdot y + p \cdot \sum_{i\in I} r_i$ by Lemma 1.8, since they only differ in the noise of the encryption and each of the r_i is chosen uniformly random from an interval exponentially larger than the noise.

□

Altogether we showed that NTRU complies with the properties required in Definition 2.4 except for active security during distributed decryption. With the methods presented in the following chapter, by Theorem 2.6 we can implement the electronic voting scheme of [CGS97] with the public key encryption scheme NTRU for a suitable choice of parameters.

4 From Passive to Active Security

In this chapter we present a variant of the protocol for secure multi-party computation given by R. Bendlin et al. in [BDOZ11] and provide an overview protocol for secure multi-party computation given by I. Damgård et al. in [DPSZ12]. Those will serve for achieving security against active adversaries during distributed decryption for protocols complying with the properties established in Section 3.1.

Using the customized version of [BDOZ11] we obtain a protocol which is secure against a static active adversaries corrupting less than half of the players. Further cheating players can be detected and discarded and afterwards the protocol restarted until only honest players remain. Additionally we give an overview on how to use the more efficient protocol presented in [DPSZ12] to achieve security against an active adversary corrupting all but one player. The disadvantage of this protocol is that cheating players cannot be detected and the protocol is not guaranteed to terminate.

Both protocols achieve security against active adversaries by letting parties authenticate their messages and thereby forcing all players to behave according to the protocol. While in [BDOZ11] the players authenticate their shares, in the more efficient approach of [DPSZ12] the secret value itself is authenticated and the authenticity checked at the end of the computation.

The protocols consist of a preprocessing phase, where keys and additional information necessary for message authentication are generated, and an online phase. The preprocessing phase can take place without knowledge of the inputs to the arithmetic circuit. In our setting this phase can thus be executed before the election begins.

For the preprocessing phases of the protocols a semi-homomorphic respectively a somewhat homomorphic cryptosystem is required. This is not to be confused with the cryptosystem used for encrypting the votes. Note that there are cryptosystems which can serve for both purposes, but in different instantiations. To avoid confusions we will refer to the underlying cryptosystem with MPKE. Further we precede the corresponding algorithms and accordingly index the corresponding variables with M.

All variables not defined explicitly are assumed to be as in Section 3.1.

4.1 Threshold Version of the BDOZ-Protocol

We will extend the protocol for secure multi-party computation given in [BDOZ11] to enable secure computation of arithmetic circuits over $R := \mathbb{F}_P[X]/\phi_N$ instead of merely $\mathbb{F}_P$. Furthermore we will use Shamir sharing instead of additive sharing to enable decryption after detecting and excluding malicious authorities. Note though that we do not consider general multi-party computation, but only computing decryption circuits as described in Section 3.1.

4.1.1 Representation of Shared Values and Underlying Cryptosystem

We first give an overview on how players can authenticate their respective shares. The generalization of message authentication from F_P to R is straightforward by authenticating each coefficient of an element in R_P separately. Therefore the terms and notations introduced in the following are very similar to the original version presented in [BDOZ11].

Additionally we describe the structure which the cryptosystem has to comply with in order to implement the preprocessing phase.

Recall that $m := m(\kappa) \in \mathbb{N}$ is the numbers of authorities in charge of the decryption and $t := t(\kappa) \in \mathbb{N}$ is chosen such that at most $t-1$ of the authorities are corrupted by a static active adversary. To be able to detect cheating authorities we have to assume an honest majority and therefore require $t - 1 < m/2$. We will work with componentwise Shamir sharing of polynomials $a \in R$ as explained in Remark 1.25.

For a value $a \in R$ each player P_i will hold a share $a_{(i)} \in R$. Additionally for each $j \in [m]\backslash\{i\}$ player P_i will hold message authentication keys $K^i_{a_{(j)}} := (\alpha^i_j, \beta^i_{a_{(j)}}) \in \mathbb{F}_P \times R$. The values $\alpha^i_j \in \mathbb{F}_P$ are chosen uniformly random for each pair of players P_i and P_j and fresh $\beta^i_{a_{(j)}} \in R$ are chosen uniformly random for each share $a_{(j)}$. We define the function

$$\mathrm{MAC}\colon R \times (\mathbb{F}_P \times R) \to R,\ (a, (\alpha, \beta)) \mapsto \alpha \cdot a + b$$

for message authentication and for each $(\alpha, \beta) \in \mathbb{F}_P \times R$ we set

$$\mathrm{MAC}_{\alpha,\beta}\colon R \to R,\ a \mapsto \mathrm{MAC}(a, (\alpha, \beta)).$$

The idea is to force players to behave according to the protocol by letting them authenticate their shares, when a value shared value is recovered.

To enable a player P_i to prove to another player P_j that he has not changed his share he gets to hold a message authentication code

$$m^j_{a_{(i)}} := \mathrm{MAC}_{\alpha^j_i, \beta^j_{a_{(i)}}}(a_{(i)}) \in R,$$

which he sends together with his share to the other players for recovering the shared value. Altogether we get the following representation of a:

$$[a] := \{a_{(i)}, \{(\alpha^i_j, \beta^i_{a_{(j)}}), m^j_{a_{(i)}}\}^m_{j=1}\}^m_{i=1}.$$

In the following for $a \in R$ we denote by $[a]$ a representation of a with variables defined as in the equation above such that $(a_{(i)})_{i\in[m]}$ indeed forms a (t, m)-Shamir sharing of a.
A representation $[a]$ is called *consistent*, if for all $i \neq j \in [m]$ it holds

$$m^i_{a_{(j)}} = \mathrm{MAC}_{(\alpha^i_j, \beta^i_{a_{(j)}})}(a_{(j)}).$$

Remark 4.1 (Unforgeability of MACs). Let $[a] := \{a_{(i)}, \{(\alpha^i_j, \beta^i_{a_{(j)}}), m^j_{a_{(i)}}\}^m_{j=1}\}^m_{i=1}$ be a consistent representation of a value $a \in R$. Let P_j be a player wanting to authenticate a value $a'_{(j)} \in R$ different of $a_{(j)}$ to P_i holding the message authentication code $m^i_{a_{(j)}}$ for the latter, and possibly $l \in \mathbb{N}$ additional values $a^1_{(j)}, \ldots, a^l_{(j)} \in R$ with corresponding MACs

$m^i_{a^1_{(j)}}, \dots, m^i_{a^l_{(j)}}$. This provides P_j with $n \cdot (l+1)$ linear equations for $n \cdot (l+1) + 1$ unknowns, thus the probability of guessing $\mathrm{MAC}_{\alpha^i_j, \beta^i_{a_{(j)}}}(a'_{(j)})$ correctly is $1/|\mathbb{F}_P| = 1/P$, which is negligible as P is super-polynomial in our setting.

In the following we describe the basic operations for jointly computing an arithmetic circuit performing mostly local operations. We omit multiplication of two private values, which is not necessary in our setting. As all proofs work similarly we will only prove that the multiplication of a constant to a shared value yields the desired consistent representation.

Opening: If $I \subseteq [m]$ is a set of at least size t, then the authorities with indices in I can jointly open a consistent representation $[a]$ as follows: For each $i \in I$ the player P_i sends a_i and $m^j_{a_i}(a_i)$ to each other player in I, who then checks if $m^j_{a_i}(a_i)$ authenticates a_i. If this is the case for all obtained values a player broadcasts "*okay*" and together with his value he can establish a by using Lagrange interpolation. Otherwise he broadcasts "*fail*".

Addition: Two consistent representations $[a]$ and $[\tilde{a}]$ of $a, \tilde{a} \in R$ can be added performing only local operations:

$$[a] + [\tilde{a}] := \{a_{(i)} + \tilde{a}_{(i)}, \{(\alpha^i_j, \beta^i_{a_{(j)}} + \tilde{\beta}^i_{a_{(j)}}), m^j_{a_{(i)}} + m^j_{\tilde{a}_{(i)}}\}^m_{j=1}\}^m_{i=1}.$$

By the properties of Shamir sharing the obtained tuple is a consistent representation of $a + \tilde{a}$.

Multiplication by scalars: If $[a]$ is a consistent representation of $a \in R$ and $\delta \in \mathbb{F}_P$ a public scalar, then

$$\delta[a] := \{\delta a_{(i)}, \{(\alpha^i_j, \delta\beta^i_{a_{(j)}}), \delta m^j_{a_{(i)}}\}^m_{i=1}\}^m_{j=1}.$$

is a consistent representation of δa.

Additions of constants: If $[a]$ is a consistent representation of $a \in R$ and $b \in R$ a public constant, then

$$b + [a] := \{b + a_{(i)}, \{(\alpha^i_j, \beta^i_{a_{(j)}} - \alpha^i_j \cdot b), m^j_{a_{(i)}}\}^m_{j=1}\}^m_{i=1}$$

is a consistent representation of $b + a$.

Multiplication of constants: If $[a]$ is a consistent representation of $a \in R$ and $b \in R$ a public constant to be multiplied to $[a]$, then

$$b \cdot [a] := \{b \cdot a_{(i)}, \{(\alpha^i_j, b \cdot \beta^i_{a_{(j)}}), b \cdot m^j_{a_{(i)}}\}^m_{j=1}\}^m_{i=1}$$

is a consistent representation of $b \cdot a$. This holds as by Remark 1.25 the values $\{b \cdot a_{(i)}\} + i \in [m]$ actually form an (t, m)-Shamir sharing of $b \cdot a$.

It is left to show that the message authentication codes for honest players are correct. By assumption the representation is consistent and thus for all $i, j \in [m]$ we have

$$m^j_{a_{(i)}} = \alpha^j_i \cdot a_{(i)} + \beta^j_{a_{(i)}}.$$

Multiplication by b on both sides yields

$$b \cdot m^j_{a_{(i)}} = \alpha^j_i \cdot b \cdot a_{(i)} + b \cdot \beta^j_{a_{(i)}}$$

and proves the claim.

In the following we will consider the message authentication keys to be elements in $\mathbb{F}_p^{1+n}$ instead of $\mathbb{F}_P \times R$. We can use the surjective coefficient embedding ι of Section 1.3 to jump between the two rings.

Before proceeding to generating the smudging values and establishing the protocol for distributed decryption we give an overview of the framework for semi-homomorphic encryption necessary for the preprocessing phase (see [BDOZ11], Section 2). The term semi-homomorphic corresponds to the requirement of satisfying a roughly additive homomorphic property as described in Definition 2.4.

Definition 4.2 ([BDOZ11] Section 2, Semi-homomorphic Encryption)**.** We call a tuple of algorithms MPKE := (MGen, MEnc, MDec) a *semi-homomorphic encryption scheme* if it complies with the following properties.

Key Generation: The algorithm MGen is randomized and takes the security parameter κ and a modulus $p \in \mathbb{N}$ as input. MGen returns a secret key sk_{M} and a public key pk_{M} and furthermore a set of parameters. More precise Gen returns natural numbers $M, R \in \mathbb{N}$ and for $d, \sigma \in \mathbb{N}$ with $\sigma < R$ the description D^d_σ of a randomized algorithm returning vectors in $\mathbb{Z}_d$ with infinity norm bounded by σ. Furthermore MGen returns an abelian group $(G, +)$. From now we will assume that every algorithm implicitly takes p, M, R, D^d_σ and G as input.

Encryption: The deterministic algorithm MEnc takes the public key pk_{M}, an integer $x \in \mathbb{Z}$ and a vector $\mathbf{r} \in \mathbb{Z}^d$ as input and returns a ciphertext $c \in G$. Furthermore for each $x_1, x_2 \in \mathbb{Z}$ and $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{Z}^d$ we require $\mathrm{MEnc}_{pk_{\mathrm{M}}}(x_1, \mathbf{r}_1) + \mathrm{MEnc}_{pk}(x_2, \mathbf{r}_2) = \mathrm{MEnc}_{pk_{\mathrm{M}}}(x_1 + x_2, \mathbf{r}_1 + \mathbf{r}_2)$. For $\tau, \rho \in \mathbb{N}$ a ciphertext $c \in G$ is called (τ, ρ)*-ciphertext* if there exists an $x \in \mathbb{Z}$ and an $\mathbf{r} \in \mathbb{Z}^d$ such that $c = \mathrm{MEnc}_{pk_{\mathrm{M}}}(x, \mathbf{r})$.

Decryption: The algorithm MDec is deterministic, takes a secret key sk_{M} and a ciphertext $c \in G$ as input and returns an element $x \in \mathbb{Z}_p \cup \{\perp\}$.

A semi-homomorphic encryption scheme MPKE with parameters as in Definition 4.2 is *correct* if there exists a negligible function negl: $\mathbb{N} \to \mathbb{R}$ such that for all $p \in \mathbb{N}$ it holds

$$\begin{aligned}&\Pr[\mathrm{MDec}_{sk_{\mathrm{M}}}(\mathrm{MEnc}_{pk_{\mathrm{M}}}(x, \mathbf{r})) \neq x \bmod p \\ &\quad \mid (pk_{\mathrm{M}}, sk_{\mathrm{M}}, M, R, D^d_\sigma, G) \leftarrow \mathrm{MGen}(1^\kappa, p), x \in \mathbb{Z}, |x| \leq M, \mathbf{r} \in \mathbb{Z}^d, \|\mathbf{r}\|_\infty \leq R] \leq \mathrm{negl}(\kappa).\end{aligned}$$

For any probabilistic polynomial time adversary $\mathcal{A}$ and $p \in \mathbb{N}$ we define the following experiment.

```
**Experiment** $Exp^{\text{IND-CPA}}_{\text{MPKE},\mathcal{A}}$ :
$(pk_{\text{M}}, sk_{\text{M}}, M, R, D^d_\sigma, G) \leftarrow \text{MGen}(1^k, p)$
$(m_0, m_1, state) \leftarrow \mathcal{A}(1^k, pk_{\text{M}})$ with $m_0, m_1 \in \mathbb{Z}_p$
$b \leftarrow \{0, 1\}$
$c := \text{Enc}_{pk}(m_b)$
$b' \leftarrow \mathcal{A}(1^k, state, c)$
**if** $b = b'$ **return** 1
**else return** 0
```

A semi-homomorphic encryption scheme MPKE is called *IND-CPA secure* if for any probabilistic polynomial time adversary $\mathcal{A}$ there exists a negligible function negl: $\mathbb{N} \to \mathbb{R}$ such that

$$\Pr\left[Exp^{\text{IND-CPA}}_{\text{MPKE},\mathcal{A}} = 1\right] \leq \frac{1}{2} + \text{negl}(\kappa).$$

Later we additionally need the notion of *admissibility* as defined in [BDOZ11], Section 4.1. For $\tau, \rho \in \mathbb{N}$ a semi-homomorphic encryption scheme MPKE is called (τ, ρ)- *admissible* if $M \geq 2^{5u+2\log u}\tau^2$ and $R \geq 2^{4u+\log u}\tau\rho$. This will ensure that the ciphertexts produced with the protocols Π_{PoPK} and π_{PoPK} given in Figure 1 and Figure 2 in [BDOZ11] for proof of plaintext knowledge and proof of correct multiplication respectively decrypt to the correct value.

4.1.2 Generation of Bounded Values and Distributed Decryption

In this section we introduce a representation for the authentication of privately generated polynomials. We will adapt the protocols of [BDOZ11] and force players to prove that the polynomials generated have bounded infinity norm. This is necessary to prevent players from adding too big smudging errors during distributed decryption and thereby tamper with the result of the decryption. Finally we present a protocol that can be used together with the cryptosystem established in Section 3.1 to obtain a protocol for distributed decryption that is secure against active adversaries corrupting less than half of the players.

For a value $a \in R$ generated and held only by player P_j we define the representation

$$[a]_j := \{a, \{m^i_a\}^m_{i=1}\}_j \cup \{(\alpha^i, \beta^i_a)\}_{i \in [m]\setminus\{j\}}$$

such that for all $i \in [m]\setminus\{j\}$ we have

$$m^i_a = \text{MAC}_{\alpha^i, \beta^i_a}(a).$$

In Figure 4.2 we give a protocol to set up bounded values of this form. For the subprotocols Π_{PoPK} and $\Pi_{2-\text{MULT}}$ we refer to Figure 1 and Figure 9 of [BDOZ11].

Functionality $\mathcal{F}_{\text{Smudge}}$

Initialization: For all indices of corrupted players $i \in I_C$ the environment specifies $\alpha_j^i \in \mathbb{F}_P$ for all indices $j \in I_H$ of honest players, i.e. $j \in I_H$. For every honest player P_i the functionality chooses for each $j \in [m]$ the value $\alpha_j^i \in \mathbb{F}_P$ uniformly random. These values will build the first components of the respective MAC-keys.

Smudging Value: On "*smudge, u*" from an honest player P_j the functionality forwards the command to the other players and proceeds as follows for all $k \in [u]$:

- If the functionality receives "*stop*" from the environment it sends "*fail*" to all players and aborts. Otherwise the functionality chooses for all $l \in [n]$ random values $a_{k,j,l} \leftarrow [-B', B']$. For all $i \in I_C$ the environment specifies values $\beta^i_{a_{k,j,l}} \in \mathbb{F}_P$.
- For all $i \in I_H \backslash \{j\}$, $l \in [n]$ the value $\beta^i_{a_{k,j,l}} \leftarrow \mathbb{F}_P$ is chosen uniformly random by the functionality.
- For all $i \in [m] \backslash \{j\}$, $l \in [n]$ the functionality sets $K^i_{a_{k,j,l}} := (\alpha^i_j, \beta^i_{a_{k,j,l}})$ and computes $m^i_{a_{k,j,l}}(a_{k,j,l}) = \alpha^i_j \cdot a_{k,j,l} + \beta^i_{a_{k,j,l}}$.
- The functionality sends the keys to the respective honest players and the MACs to P_j.

On "*smudge, u*" from a player P_j controlled by the adversary the functionality forwards the command to the other players and proceeds as follows for all $k \in [u]$:

- If the functionality receives "*stop*" from the environment it sends "*fail*" to all players and aborts. Otherwise the environment specifies for all $l \in [n]$ values $a_{k,j,l} \in [-B', B']$ and $m^i_{a_{k,j,l}} \in \mathbb{F}_P$ for each $i \in [m]$.
- For all $i \in [m] \backslash \{j\}$ and $l \in [n]$ the functionality computes the values $\beta^i_{a_{k,j,l}} := m^i_{a_{k,j,l}} - \alpha^i_j \cdot a_{k,j,l}$.
- For all $i \in [m] \backslash \{j\}$, $l \in [n]$ the functionality sets $K^i_{a_{k,j,l}} := (\alpha^i_j, \beta^i_{a_{k,j,l}})$.
- The functionality sends the keys to the respective honest players.

Figure 4.1: Ideal Functionality for the Generation of Smudging Values (compare [BDOZ11], Figure 5)

Theorem 4.3 (Compare [BDOZ10], Theorem 2). *The protocol* Π_{Smudge} *(Figure 4.2) implements the ideal functionality* $\mathcal{F}_{\text{Smudge}}$ *(Figure 4.1) in the* $(\mathcal{F}_{\text{KeyReg}})$*-hybrid model ([BDOZ10], Figure 9) securely against any static active adversary corrupting up to* $t-1$ *authorities if the cryptosystem* MPKE = (MGen, MEnc, MDec) *is semi-homomorphic modulo* P*,* (B', ρ)*-admissible and* IND-CPA *secure.*

Proofsketch. In the following we present a simulator $\mathcal{S}_{\text{Smudge}}$ and show that an arbitrary environment $\mathcal{Z}$ cannot decide whether it is interacting with the real protocol Π_{Smudge} or the simulator $\mathcal{S}_{\text{Smudge}}$ on top of the ideal functionality $\mathcal{F}_{\text{Smudge}}$.

We can assume without loss of generality that the scheme MPKE has additionally to the key generation algorithm MGen a randomized algorithm MGen* which outputs a meaningless public key $pk_{\text{M}}^{\star}$ computationally indistinguishable from a public key generated by MGen and such that the encryption of any message under $pk_{\text{M}}^{\star}$ is statistically indistinguishable from an encryption of 0. For the proof we refer to Section 4.3 of [BDOZ10].

The underlying idea of the simulator $\mathcal{S}_{\text{Smudge}}$ is to simulate calls to the ideal functionalities $\mathcal{F}_{\text{KeyReg}}$ and run a copy of the protocol Π_{Smudge} internally. The simulator thereby learns the shares and values of authorities corrupted by the adversary - as he knows the secret keys - and can give them as input to the functionality $\mathcal{F}_{\text{Smudge}}$. In the following we give a precise Definition (compare [BDOZ10], Figure 16).

Initialization: For each $i \in [m]$ the player P_i chooses values $\alpha_1^i, \ldots, \alpha_m^i \leftarrow \mathbb{F}_P$ uniformly random, publishes encryptions of those values under his respective public key and uses the protocol Π_{PoPK} ([BDOZ11], Figure 1) to convince player P_j that $\text{Enc}_{pk_{\text{M}}^i}(\alpha_j^i)$ is a valid encryption of a message in $\mathbb{F}_P$ (each player P_i holds a secret key sk_{M}^i and publishes the according public key pk_{M}^i).

Simulator $\mathcal{S}_{\text{Smudge}}$:

Initialization: The simulator invokes MGen(1^{κ}) several times to generate key pairs $(sk_{\text{M}}^i, pk_{\text{M}}^i)$ for each $i \in [m]$ and sends all public keys and for each $i \in I_C$ the corresponding shares of the secret key sk_{M}^i to the respective party corrupted by the adversary. Then the simulator proceeds with the initialization step of the protocol Π_{Smudge}, where it plays the role of the honest players. If the protocol Π_{PoPK} fails for a pair of players including an honest player, the simulator aborts. For every pair of a corrupt player P_i and an honest player P_j the simulator can recover α_j^i by decrypting the obtained encryption, as it has knowledge of all secret keys. Finally the simulator can forward the obtained values as input of the initialization step to the ideal functionality $\mathcal{F}_{\text{Smudge}}$.

Smudging Value: The simulator executes the smudging function of Π_{Smudge} playing the role of the honest players. If the subprotocols Π_{PoPK} or Π_{PoCM} fail at any step during execution for a pair of players including an honest player, the simulator sends "*stop*" to the functionality and aborts. We will take a look at two scenarios separately. First we assume that the player P_j invoking the protocol is honest. In this case the simulator proceeds as follows for each $k \in [u]$:

- The simulator executes the function for generating a smudging value of the protocol Π_{Smudge} playing the role of P_j and the other honest players.
- The value $\beta_{a_{k,j,l}}^i$ for each corrupt player P_i and each $l \in [n]$ can be calculated by the simulator, as he has knowledge of α_j^i because of the initialization step and gets knowledge of the corresponding MAC value during the last step of a successful execution of Π_{Smudge}.

Protocol Π_{Smudge}

Smudging Value(j, u)**:**

- Player P_j chooses for all $k \in [u]$, $l \in [n]$ a value $a_{k,j,l} \leftarrow [-B', B']$ and sends $\{\mathrm{Enc}'_{pk^j_{\mathrm{M}}}(a_{k,j,l})\}_{k\in[u],l\in[n]}$ using randomness bounded by ρ to the other players.
- Player P_j runs with every other player P_i the Σ'-protocol $\Pi_{\mathrm{PoPK}}(u, B', \rho)$ with the ciphertexts created during the first step as input to prove that the messages are bounded by $2^{2u+\log u}B'$ and the used randomness by $2^{2u+\log u}\rho$.
- For every $i \in [m]\backslash\{j\}$ player P_i and P_j invoke $\Pi_{2-\mathrm{MULT}}(u, B', \rho)$ ([BDOZ11], Figure 9) with input $\{\mathrm{MEnc}_{pk^i_{\mathrm{M}}}(\alpha^i_j)\}_{k\in[u],l\in[n]}$ and $\{\mathrm{MEnc}_{pk^j_{\mathrm{M}}}(a_{k,j,l})\}_{k\in[u],l\in[n]}$. Let $\{z_{k,i,l}\}_{k\in[u],l\in[n]}$ denote the output of P_i and $\{z_{k,j,l}\}_{k\in[u],l\in[n]}$ the output of P_j. The result is for each $k \in [u]$ a representation

$$[a_{k,j}]_j := \{a_{k,j}, \{z_{k,i}\}_{i=1}^m\}_j \cup \{(\alpha^i_j, -z_{k,j}))\}_{i\in[m]\backslash\{j\}},$$

where $a_{k,j} := \sum_{l=1}^n a_{k,j,l}X^{l-1} \in R$ and $z_{k,\iota} := (z_{k,\iota,l})_{l\in[n]} \in \mathbb{F}_p^n$ for $\iota \in \{i, j\}$.

Figure 4.2: Protocol for Constructing Verifiable Bounded Polynomials (compare [BDOZ11], Figure 12)

In case the player P_j is controlled by the adversary the simulator proceeds as follows for each $k \in [u]$:

- The simulator executes the function for generating a smudging value of the protocol Π_{Smudge} playing the role of the honest players. He decrypts all the messages send by P_j during the first step, thereby obtaining the values $(a_{k,j,l})_{l\in[n]}$.
- For each honest player P_i he obtains for all $k \in \{t, \ldots, m\}$, $l \in [n]$ the keys $K^i_{a_{k,j,l}}$ during the third step and can thus calculate the corresponding MAC values.

In both cases finally the simulator calls the function for generating a smudging value on the ideal functionality with the inputs and shares of the corrupted parties acquired previously. The idea now is to show security by a reduction argument, namely by showing that a distinguisher between the real and simulated view can be used to distinguish between a real public key generated by MGen and a meaningless public key generated by MGen*, and thus proving that the two views are computationally indistinguishable, see [BDOZ10], Section 4.3 for more details.

□

Now we present the protocol for active secure threshold decryption. The protocol is obtained by using the protocol Π_{PDec} and additionally checking the values returned by each

Functionality $\mathcal{F}_{\text{KeyGenMAC}}$

- Upon receiving "*start*" from all honest players, run the algorithm for key generation $(pk, sk) \leftarrow \text{Gen}(1^\kappa)$ and send pk to the adversary.
- For each $i \in I_C$ receive for each $k \in \{t, \ldots, m\}$ a share $sk^k_{(i)} = \left(sk^k_{(i),l}\right)_{l \in [s_2]} \in (\mathbb{F}_P[X]/\phi_N)^{s_2}$ and additionally for each $l \in [s_2]$ and $j \in I_H$ values $m^j_{k,i,l} \in \mathbb{F}^n_P$ and $K^i_{k,j,l} := \left(\alpha^i_j, \beta^i_{k,j,l}\right) \in \mathbb{F}_P \times \mathbb{F}^n_P$ from the adversary.
- Calculate for each honest player P_j for each $k \in \{t, \ldots, m\}$ shares $sk^k_{(j)} \in S_{sk}$ as in Remark 1.25 such that $\left(sk^k_{(j)}\right)_{j \in [m]} \in (\mathbb{F}_P[X]/\phi_N)^{s_2}$ forms a (k, m)-Shamir sharing of the secret key. For each $i \in I_C$, $l \in [s_2]$ the simulator further calculates $m^i_{sk^k_{(j),l}} := \alpha^i_j \cdot sk^k_{(j),l} + \beta^i_{k,j,l}$. For each pair of honest player P_j and dishonest player P_i the simulator chooses $\alpha^j_i \leftarrow \mathbb{F}_P$ uniformly random and sets $\beta^j_{sk^k_{(i),l}} := \alpha^j_i \cdot sk^k_{(i),l} - m^j_{k,i,l}$. For each pair of honest players P_i, P_j the key $\left(\alpha^j_i, \beta^j_{sk^k_{(i),l}}\right) \leftarrow \mathbb{F}_P \times \mathbb{F}^n_P$ is chosen uniformly random and the MAC value calculated.
- Send the public key pk to each player and additionally to each honest player P_i the set of shares $\left(sk^k_{(i)}\right)_{k \in \{t,\ldots,m\}}$ and the respective keys and MACs.

Figure 4.3: Ideal Functionality for Key Generation with Authentication Codes

player with the help of message authentication codes. In this way players not following the protocol can be detected and excluded. As described in Section 3.1 each authority P_i is assumed to hold a set of secret keys $(sk^k_{(i)})_{k \in \{t,\ldots,m\}}$ after key distribution, where $sk^k_{(i)} \in R^{s_2}$ for each $k \in \{t, \ldots, m\}$. Let C be the decryption circuit with input $sk^k_{(i)} \in R^{s_2}$ and $c \in \mathcal{C}$ and output in R as specified in Section 3.1.

The protocol for obtaining the final results will consist of several rounds, where after each round cheating players are excluded. It terminates as soon as every player behaves according to the protocol, that is after at most $t - 1 < m/2$ rounds. Let $I \subseteq [m]$ be the index set of players participating at each round and $k \in \mathbb{N}$ be the round number. In the first round we have $I = [m]$ and $k = 1$.

Theorem 4.4. *The protocol Π_{ADec} (Figure 4.5) implements the ideal functionality $\mathcal{F}_{\text{ADec}}$ (Figure 4.4) in the $(\mathcal{F}_{\text{Smudge}}, \mathcal{F}_{\text{KeyGenMAC}})$-hybrid model (Figure 4.1, 4.3) with statistical security against any active static adversary corrupting up to $t - 1$ parties, assuming that the underlying protocol Π_{PDec} implements the ideal functionality $\mathcal{F}_{\text{PDec}}$ (Figure 3.2) in the $\mathcal{F}_{\text{KeyGen}}$-hybrid model (Figure 3.1) with statistical security against any active static adversary corrupting up to $t - 1$ parties.*

Functionality $\mathcal{F}_{\text{ADec}}$

- Upon receiving "*start*" from all honest players, run the algorithm for key generation $(pk, sk) \leftarrow \text{Gen}(1^\kappa)$, send pk to the adversary and store sk.
- Upon receiving "*decrypt c*" from all honest players or in the Shamir setting "*decrypt c, I*" from a set $I \subseteq [m]$ of at least t players respectively, send c and $v := \text{Dec}_{sk}(c)$ to all players with index in $[m]$ or in I respectively and the adversary.

Figure 4.4: Ideal Functionality for Distributed Decryption (compare [BD10], Section 4.3)

Proofsketch. (Compare [BDOZ10], Theorem 1) We need to give a simulator $\mathcal{S}_{\text{ADec}}$ and show that any environment polynomially bounded environment $\mathcal{Z}$ cannot decide whether it is interacting with the real protocol Π_{ADec} with access to the functionalities $\mathcal{F}_{\text{KeyGen}}$ and $\mathcal{F}_{\text{Smudge}}$ or the simulator $\mathcal{S}_{\text{ADec}}$ on top of the functionality $\mathcal{F}_{\text{ADec}}$. By our assumption there exists a simulator $\mathcal{S}_{\text{PDec}}$, such that any environment $\mathcal{Z}$ cannot decide whether it is interacting with Π_{PDec} or with S_{PDec} on top of the ideal functionality $\mathcal{F}_{\text{PDec}}$. The simulator $\mathcal{S}_{\text{ADec}}$ then will proceed as follows:

Simulator $\mathcal{S}_{\text{ADec}}$:

Key Generation: When receiving "*start*" the simulator $\mathcal{S}_{\text{ADec}}$ forwards the command to the ideal functionality $\mathcal{F}_{\text{KeyGenMAC}}$ and obtains the public key pk from the functionality. Furthermore for each $i \in I_C$ the simulator receives for each $k \in \{t, \dots, m\}$ a share $sk^k_{(i)} = (sk^k_{(i),l})_{l\in[s_2]} \in (\mathbb{F}_P[X]/\phi_N)^{s_2}$ and additionally for each $l \in [s_2]$ and $j \in I_H$ values $m^j_{k,i,l} \in \mathbb{F}^n_P$ and $K^i_{k,j,l} \in \mathbb{F}^{1+n}_P$. The simulator stores the latter and proceeds like the simulator $\mathcal{S}_{\text{PDec}}$ with the shares of the secret key.

Initialization: The simulator internally runs a copy of $\mathcal{F}_{\text{Smudge}}$ and creates the desired number of smudging values. He reads and stores all data of the parties corrupted by the adversary.

Decryption:

- For the players corrupted by the adversary the simulator $\mathcal{S}_{\text{ADec}}$ received the shares and the message authentication codes during the key generation and can therefore for all $i \in I_C$ simulate the calculation of r_i and the corresponding message authentication codes.
- For each $i \in I_C$ the simulator $\mathcal{S}_{\text{ADec}}$ calculates $r'_{(i)}$. For each $j \in I_H$ the value $r'_{(j)}$ is calculated by the simulator $\mathcal{S}_{\text{PDec}}$. In both cases the corresponding message authentication codes can be calculated with the information received and stored during the initialization step.

Protocol Π_{ADec}

Initialization: The authorities invoke $\mathcal{F}_{\text{Smudge}}(i,u)$ for each $i \in [m]$ once (or in case $u < t$ then $\lceil t/u \rceil$ times) to obtain representations of smudging values $(a_{k,i})_{k\in[t]}$.

Decrypt(c, I, k)**:**

- For each $i \in I$ the player P_i calculates $r_{(i)} := C(sk_{(i)}^{|I|}, c) \in \mathbb{F}_P[X]/\phi_N$ and all players update the keys and message authentication codes accordingly.
- For every $i \in I$ we set $d_i := d_{i,I}$ as defined in Remark 1.24 and player P_i calculates $r'_{(i)} := r_{(i)} + d_i \cdot a_{k,i} \in \mathbb{F}_P[X]/\phi_N$ and for all $j \in I\backslash\{i\}$ the new key $\beta^i_{r'_{(j)}} := \beta^i_{r_{(j)}} + d_j \cdot \beta^i_{a_{k,j}} \in \mathbb{F}_P^n$ and the new message authentication code $m^j_{r'_{(i)}} := m^j_{r_{(i)}} + d_j \cdot m^j_{a_{k,i}} \in \mathbb{F}_P^n$.
- Now the players invoke Opening(I) for opening the value r'. If all players succeed the obtained value is the result plus the smudging value $\sum_{i\in I} a_{k,i}$. Otherwise the players continue with the next step.
- Let I' be the set of indices of all players for which the MAC-check succeeded at least $m - t$ times. Let $k' := k + 1$. Then the players of I' invoke $\mathrm{Decrypt(c, I', k')}$.

Figure 4.5: Protocol for Actively Secure Threshold Decryption

- During the opening step the simulator has to check the received message authentication codes for messages honest players received from corrupted players and simulate the messages and MACs sent by honest players. This can be done with the information gathered in previous steps.

The simulator aborts in case the simulated protocol fails during the execution because of a wrong message authentication code. If the protocol execution is successful the computation is correct with overwhelming probability, because the success probability of a player to forging a value for a given message authentication code is at most $1/u$. Since the MAC checks of honest players only possibly fail for corrupted players, the MAC check of each honest player succeeds at least $m - t$ times, ensuring that honest players are not excluded from the protocol. $\square$

4.2 SPDZ-Protocol

In this section we give a short overview of the so called SPDZ-protocol for secure multi-party computation presented in [DPSZ12]. Using this protocol for distributed decryption we obtain active security against up to $m - 1$ malicious authorities. The other major advantage of this approach over the one presented in the previous section is the boost in efficiency: Previously for every pair of players and each share a separate key was necessary for message authentication. In the new approach the secret values themselves are authenticated with one global key instead. A disadvantage is that there is no guarantee that the protocol terminates and malicious players cannot be identified.

In contrast to Section 4.1 we will use the notion of [DPSZ12] directly, that means we will work with a protocol for secure multi-party computation over $\mathbb{F}_P$ and not $\mathbb{F}_P[X]/\phi_N$. This approach does not yield any restrictions, as all necessary calculations over $\mathbb{F}_P[X]/\phi_N$ can be performed coefficientwise.

A crucial point of the SPDZ-protocol is that it uses a single global key, giving an authority knowledge of this key would allow to forge message authentication codes. Then again players cannot check the validity of message authentication codes without having knowledge of the global key. This seeming contradiction is resolved by postponing checking the authenticity of values to the end and only do partial openings during the ongoing computation. This may result in wrong intermediary results, but the final check will ensure that cheating is detected. The global key will be shared among the players and before it is opened at the end, each player has to commit to the results, thereby preventing corrupted players to exploit knowledge of the global key.

4.2.1 Representation of Shared Values and Underlying Cryptosystem

We now give an overview of the representation of values, of the authentication check at the end of the computation and requirements on the public key encryption scheme necessary for preprocessing (see [DPSZ12], Section 2 and 3).

Let $a \in \mathbb{F}_P$ be an arbitrary value and $\alpha \in \mathbb{F}_P$ the global key. Then the *representation* of a sharing of a between m authorities consists of a public variable $\delta_a \in \mathbb{F}_P$, for each $i \in [m]$ a value a_i privately held by P_i such that $a = \sum_{i=1}^m a_i$ and furthermore a value $\gamma_{a,i} \in \mathbb{F}_P$ such that $\sum_{i=1}^m \gamma_{a,i} = \alpha(a + \delta_a)$. In this case we write

$$\langle a \rangle = (\delta_a, (a_i)_{i\in[m]}, (\gamma_{a,i})_{i\in[m]}).$$

Addition of shared values and *multiplication of constants* can be done locally componentwise. Let

$$\langle b \rangle := (\delta_b, (b_i)_{i\in[m]}, (\gamma_{b,i})_{i\in[m]})$$

be a representation of $b \in \mathbb{F}_P$ and further let $e \in \mathbb{F}_P$. Then

$$(\delta_a + \delta_b, (a_i + b_i)_{i\in[m]}, (\gamma_{a,i} + \gamma_{b,i})_{i\in[m]})$$

is a representation of $a + b$ and

$$(e\delta_a, (ea_i)_{i\in[m]}, (e\gamma_{a,i})_{i\in[m]})$$

is a representation of ea. *Addition of constants* can be performed locally with the help of the public constant, namely

$$(\delta_a - e, (a_i')_{i\in[m]}, (\gamma_{a,i})_{i\in[m]})$$

with $a_1' := a_1 + e$ and $a_i' := a_i$ for $i \in [m]\backslash\{1\}$ is a representation of $e + a$. For the protocol for *multiplication of shared values* we refer to [DPSZ12], Figure 1.

Before proceeding to the preprocessing phase, we give an idea on how the authenticity of the final value is checked. As brought up earlier the checking is postponed until after the end of the evaluation. During the computation itself all values are only partially opened and no message authentication codes are checked. For the global key $\alpha \in \mathbb{F}_p$ a different representation is used, namely

$$[\![\alpha]\!] := ((\alpha_i)_{i\in[m]}, (\beta_i, \gamma^i_{\alpha,1}, \cdots \gamma^i_{\alpha,m})_{i\in[m]}),$$

where all variables are elements in $\mathbb{F}_p$ and player P_i holds $\alpha_i, \beta_i, \gamma^i_{\alpha,1}, \cdots \gamma^i_{\alpha,m}$. Further $\alpha = \sum_{i=1}^m \alpha_i$ and for all $j \in [m]$ it holds $\sum_{i=1}^m \gamma^i_{\alpha,j} = \alpha\beta_j$. The intuition of the latter equation is, that the value on the left-hand side is the message authentication code authenticating the global key α to P_j owning the private key β_j.

The output phase now proceeds as follows, where $\mathcal{F}_{\text{Com}}$ is assumed to be an ideal functionality for commitments. Assume that the output value $y \in \mathbb{F}_p$ is shared by the players, but not opened yet. Further let $T \in \mathbb{N}$ and assume that the values $(a_j)_{j\in[T]}$ with $\langle a_j\rangle = (\delta_j, (a_{j,i})_{i\in[m]}, (\gamma_{a_j,i})_{i\in[m]}$ for all $i \in [T]$ were partially opened and not checked during the protocol execution so far. Let $[\![e]\!]$ be a random value produced during preprocessing (see [DPSZ12], Figure 7). Then the parties proceed as follows (see [DPSZ12], Figure 1).

1. The players open the random value $[\![e]\!]$ and compute $a := \sum_{j=1}^T e^j a_j$.

2. Each player P_i computes $\gamma_i := \sum_{j=1}^T e^j \gamma_{a_j,i}$.

3. Each player P_i calls $\mathcal{F}_{\text{Com}}$ to commit to γ_i, his share y_i of y and the corresponding message authentication code $\gamma_{y,i}$.

4. The players open the global key $[\![\alpha]\!]$.

5. The ideal functionality $\mathcal{F}_{\text{Com}}$ opens γ_i for each $i \in [m]$. Each player checks whether $\alpha\left(a + \sum_{j=1}^T e^j \delta_j\right) = \sum_{i=1}^m \gamma_i$ and sends "*okay*" if the equality holds. If all player send "*okay*" the protocol proceeds to the last step. Otherwise the protocol aborts.

6. The ideal functionality $\mathcal{F}_{\text{Com}}$ opens y_i and $\gamma_{y,i}$ for each $i \in [m]$. Now y can be recovered as $\sum_{i=1}^m y_i$ and each player checks whether $\alpha(y + \delta_y) = \sum_{i=1}^m \gamma_{y,i}$. If this check is successful y is the output. Otherwise the protocol aborts.

The underlying idea of this protocol is that the commitments prevent corrupted players to exploit knowledge of the global key and change their authentication codes accordingly. Note that in [DKL+13], Figure 3 an improvement of this protocol is given, where players do not have to reveal their message authentication keys, thereby allowing to reuse preprocessed data.

For the preprocessing phase of the protocol for secure multi-party computation in [DPSZ12] a encryption scheme is needed. We explain the basic properties a cryptosystem MPKE := (MParamGen, MKeyGen, MKeyGen*, MEnc, MDec) has to comply with for this purpose (see [DPSZ12], Section 5). Basically a certain number of homomorphic additions and one homomorphic multiplication has to be supported. Such an encryption scheme is called *somewhat homomorphic*. We will however not explain all details, for more information we refer to Section 3 in [DPSZ12].

Message Space: The message space is of the form $\mathcal{M}_\mathrm{M} := (\mathbb{F}_P)^{s_\mathrm{M}}$ for some $s_\mathrm{M} \in \mathbb{N}$, that is a direct product of finite fields with componentwise addition and multiplication. Furthermore there exists an injective function

$$\text{encode}\colon \mathcal{M}_\mathrm{M} \hookrightarrow \mathbb{Z}^N$$

for a suitable natural number $N \in \mathbb{N}$ and a function

$$\text{decode}\colon \mathbb{Z}^N \to \mathcal{M}_\mathrm{M}$$

with the following properties:

1. For all $\mathbf{m} \in \mathcal{M}_\mathrm{M}$ it holds $\|\text{encode}(\mathbf{m})\|_\infty \leq P/2$.
2. For all $\mathbf{m} \in \mathcal{M}_\mathrm{M}$ it holds $\text{decode}(\text{encode}(\mathbf{m})) = \mathbf{m}$.
3. For all $\mathbf{x} \in \mathbb{Z}^N$ it holds $\text{decode}(\mathbf{x}) = \text{decode}(\mathbf{x} \bmod P)$.
4. For all $\mathbf{m}_1, \mathbf{m}_2 \in \mathcal{M}_\mathrm{M}$ it holds $\text{decode}(\text{encode}(\mathbf{m}_1) + \text{encode}(\mathbf{m}_2)) = \mathbf{m}_1 + \mathbf{m}_2$ and $\text{decode}(\text{encode}(\mathbf{m}_1) \cdot \text{encode}(\mathbf{m}_2)) = \mathbf{m}_1 \cdot \mathbf{m}_2$, where $+$ is componentwise addition and $\cdot$ an arbitrary operation on $\mathbb{Z}^N$.

Space of Randomness: Let $d, \rho \in \mathbb{N}$ be natural numbers. The space of randomness is $\mathbb{Z}^d$. By D^d_ρ we denote a randomized algorithm which outputs elements in $\mathbb{Z}^d$ with infinity norm bounded by ρ with overwhelming probability.

Ciphertext Space: The ciphertext space is an abelian group $(\mathcal{G}, \boxplus)$. Additionally$(\mathcal{G}, \boxplus)$ has to support a not necessarily closed multiplicative operation $\boxtimes$, which is commutative and furthermore distributive with $\boxplus$.

Parameter Generation: The probabilistic algorithm MParamGen takes 1^κ and the message space $\mathcal{M}$ as input and returns all parameters introduced above and furthermore a set C of allowable arithmetic SIMD circuits over $\mathcal{M}$.

Key Generation: The probabilistic algorithm MKeyGen returns a pair of keys, a public key $pk_\mathrm{M} \in S_{pk_\mathrm{M}}$ and a secret key $sk_\mathrm{M} \in S_{sk_\mathrm{M}}$.

Encryption: The deterministic algorithm $\mathrm{MEnc}_{pk_\mathrm{M}}$ takes an encoded message $\mathbf{x} \in \mathbb{Z}^N$ and a vector of randomness $\mathbf{r} \in \mathbb{Z}^d$ and returns a ciphertext $c \in \mathcal{G}$. If the randomness is not specified, it is assumed to be drawn according to the distribution D^d_ρ.

Decryption: The deterministic algorithm $\mathrm{MDec}_{sk_\mathrm{M}}$ takes a ciphertext $c \in \mathcal{G}$ and returns a message $\mathbf{m} \in \mathcal{M}$ or $\perp$.

Correctness: Let $B_\text{plain}, B_\text{rand} \in \mathbb{N}$ be natural numbers with $P/2 \leq B_\text{plain}$ and $\rho \leq B_\text{rand}$ and C the set of allowed arithmetic SIMD circuits over $\mathcal{M}$. Then we say the cryptosystem is $(B_\text{plain}, B_\text{rand}, C)$*-correct* if there exists a negligible function $\text{negl}\colon \mathbb{N} \to R_{\geq 0}$

such that for each $f \in C$ and for each $\mathbf{x} \in \mathbb{Z}^N$, $\mathbf{r} \in \mathbb{Z}^d$ with $||\mathbf{x}||_\infty \leq B_{\text{plain}}$, decode$(\mathbf{x}) \in \mathcal{M}$ and $||\mathbf{r}||_\infty \leq B_{\text{rand}}$ it holds

$$\Pr\Big[\text{MDec}_{sk_{\text{M}}}(\hat{f}(\text{MEnc}_{pk_{\text{M}}}(\mathbf{x},\mathbf{r}))) \neq f(\text{decode}(\mathbf{x})) \\ \mid P \leftarrow \text{MParamGen}(1^\kappa, \mathcal{M}), (pk_{\text{M}}, sk_{\text{M}}) \leftarrow \text{MKeyGen}()\Big] \leq \text{negl}(\kappa),$$

where $\hat{f}$ is the function on $\mathcal{G}$ induced by f obtained by replacing the operations on $\mathcal{M}$ with the corresponding ones on $\mathcal{G}$ and by replacing every constant $\mathbf{m} \in \mathcal{M}$ with $\text{MEnc}_{pk_{\text{M}}}(\mathbf{m}, \mathbf{0})$.

Admissibility: Let each formula $f \in C$ be of the form $(x_1 + \cdots + x_m) \cdot (y_1 + \cdots + y_m) + z_1 + \cdots + z_n$ or consisting of less additions and/or no multiplication. Then the cryptosystem is called *admissible*, if there exists a $v \in \mathbb{R}_{>0}$ such that the cryptosystem is $(B_{\text{plain}}, B_{\text{rand}}, C)$-correct for

$$B_{\text{plain}} := N \cdot P/2 \cdot u^2 \cdot 2^{(1/2+v)\cdot u} \text{ and } B_{\text{rand}} := d \cdot \rho \cdot u^2 \cdot 2^{(1/2+v)\cdot u},$$

where $u \in \mathbb{N}$ is the statistical security parameter.

4.2.2 Generation of Bounded Values

To define the protocol for generating smudging values, we have to provide the authorities with a protocol for proving that their respective encrypted message is bounded. To be more precise we will give a Σ' protocol for the relations

$$\begin{aligned} R_{PoBPK} := \{(x, w) \mid & x = (pk, \mathbf{c}) \in S'_{pk} \times (C')^u \text{ with } \mathbf{c} = (c_i)_{i\in[u]} \\ & \wedge w = (\mathbf{x}_i, \mathbf{r}_i)_{i\in[u]} \in (\mathcal{M}' \times \mathcal{U}')^u \\ & \wedge \forall i \in [u]\colon c_i = \text{MEnc}_{pk}(\mathbf{x}_i, \mathbf{r}_i),\ \text{decode}(\mathbf{x}_i) \in \mathbb{F}_p^{s'} \\ & \wedge \forall i \in [u]\colon ||\mathbf{x}_i||_\infty \leq B,\ ||\mathbf{r}_i||_\infty \leq \rho\} \end{aligned}$$

and

$$\begin{aligned} R'_{PoBPK} := \{(x, w) \mid & x = (pk, \mathbf{c}) \in S'_{pk} \times (C')^u \text{ with } \mathbf{c} = (c_i)_{i\in[u]} \\ & \wedge w = (\mathbf{x}_i, \mathbf{r}_i)_{i\in[u]} \in (\mathcal{M}' \times \mathcal{U}')^u \\ & \wedge \forall i \in [u]\colon c_i = \text{MEnc}_{pk}(\mathbf{x}_i, \mathbf{r}_i),\ \text{decode}(\mathbf{x}_i) \in \mathbb{F}_p^{s'} \\ & \wedge \forall i \in [u]\colon ||\mathbf{x}_i||_\infty \leq B',\ ||\mathbf{r}_i||_\infty \leq B_{\text{rand}}\}, \end{aligned}$$

where $B \in \mathbb{N}$ with $B \leq \tau$ suitable and $B' := N \cdot B \cdot u^2 \cdot 2^{(1/2+v)u}$. From now on let $\tilde{u} := u^2 \cdot 2^{v \cdot u - 1}$. The desired protocol Π_{PoBPK} can be found in Figure 4.6. It is a variant of the protocol Π_{ZKPoPK} which can be found in the full version of [DPSZ12], Appendix A.1. We omit the proof of the following Lemma, as is works analogous to the proof of Theorem 5 given there.

Lemma 4.5. *Define all parameters as above. The protocol* Π_{PoBPK} *(Figure 4.6) is a* Σ'*-proof for the relations* $R_{\text{PoBPK}} \subseteq R'_{\text{PoBPK}}$.

Protocol Π_{PoBPK}

$x = (pk, \mathbf{c}),\ w = (\mathbf{x}_i, \mathbf{r}_i)_{i \in [u]}$ $\downarrow$ **Prover** P

$x = (pk, \mathbf{c})$ $\downarrow$ **Verifier** V

Prover P:

for all $i \in [V]$:
- $\mathbf{m}_i \leftarrow (\mathbb{F}_P)^{s'}$ such that $\|\operatorname{encode}(\mathbf{m}_i)\|_\infty \leq B$ and
- $\mathbf{u}_i \leftarrow (p\mathbb{Z})^N$ such that $\operatorname{encode}(\mathbf{m}_i) + \mathbf{u}_i \in (I_1)^N$,
- $\mathbf{y}_i := \operatorname{encode}(\mathbf{v}_i) + \mathbf{u}_i$,
- $\mathbf{s}_i \leftarrow (I_2 \cap \mathbb{Z})^d$

$$\xrightarrow{\mathbf{a} := (\mathrm{MEnc}_{pk}(\mathbf{y}_i, \mathbf{s}_i))_{i \in [V]}}$$

Verifier V: $\mathbf{e} \leftarrow \{0,1\}^u$

$$\xleftarrow{\mathbf{e}}$$

Prover P:

$Y := (\mathbf{y}_1, \ldots, \mathbf{y}_V) \in (\mathbb{Z}^N)^V$

$X := (\mathbf{x}_1, \ldots, \mathbf{x}_u) \in (\mathbb{Z}^N)^u$

$S := (\mathbf{s}_1, \ldots, \mathbf{s}_V) \in (\mathbb{Z}^d)^V$

$R := (\mathbf{r}_1, \ldots, \mathbf{r}_u) \in (\mathbb{Z}^d)^u$

$$\xrightarrow[T := S^\top + M_{\mathbf{e}} \cdot R^\top]{Z := Y^\top + M_{\mathbf{e}} \cdot X^\top}$$

Verifier V:

for all $i \in [V]$:
- $\mathbf{z}_i := Z_i,\ \mathbf{t}_i := T_i$,
- $d_i := \mathrm{MEnc}_{pk}(\mathbf{z}_i, \mathbf{t}_i)$,
- $\operatorname{decode}(\mathbf{z}_i) \overset{?}{\in} (\mathbb{F}_P)^{s'}$

$\mathbf{d} := (d_1, \ldots, d_V) \in \mathcal{G}^V$,

$\mathbf{d}^\top \overset{?}{=} \mathbf{a}^\top \boxplus (M_{\mathbf{e}} \boxtimes \mathbf{c}^\top)$,

$Z \overset{?}{\in} (I_1 \cap \mathbb{Z})^{V \times N}$,

$T \overset{?}{\in} (I_2 \cap \mathbb{Z})^{V \times d}$

where $V := 2 \cdot u - 1 \in \mathbb{N}$ is the number of random ciphertexts to be constructed by the prover, $I_1 := [-B \cdot N \cdot \tilde{u}, B \cdot N \cdot \tilde{u}]$ and $I_2 := [-\rho \cdot d \cdot \tilde{u}, \rho \cdot d \cdot \tilde{u}]$ are intervals, for an arbitrary matrix M by M_i we denote the i-th row of M and further we let $M^{\mathbf{e}} := (m_{i,j}^{\mathbf{e}})_{i \in [V], j \in [u]} \in \mathbb{Z}^{V \times u}$ be the matrix with entries

$$m_{i,j}^{\mathbf{e}} := \begin{cases} \mathbf{e}_{i-j+1}, & \text{if } 1 \leq i - j + 1 \leq u \\ 0, & \text{otherwise} \end{cases}$$

Figure 4.6: Protocol for Proving Boundedness of an Encrypted Message (compare [DPSZ12] (Full Version), Appendix A.1)

Smudging Value: The result of this function is a shared value guaranteed to be bounded by $B' := N \cdot B \cdot u^2 \cdot 2^{(1/2+\nu)u}$ for $B \in \mathbb{N}$ suitable.

- For each $i \in [m]$ player P_i generates $r_i \in \mathbb{F}_P$, such that $|r_i| \leq B/m$. Let $r := \sum_{i=1}^{m} r_i \in \mathbb{F}_P$.
- For each $i \in [m]$ player P_i encrypts the value r_i and broadcasts the encryption. That means he sends $c_i := \mathrm{MEnc}_{pk}(r_i)$ to the other players. Let $c := \boxplus_{i=1}^{m} c_i$.
- For each $i \in [m]$ player P_i uses Π_{PoBPK} to prove that the ciphertext c_i he generated encrypts a plaintext with absolute value bounded by B'/m.
- Players generate $\langle r \rangle \leftarrow \mathrm{PAngle}(r_1, \ldots, r_n, c)$ ([DPSZ12], Figure 6).

Figure 4.7: Protocol for Generating Smudging Values (compare [DPSZ12], Figure 7)

Now it is left to complement the protocol presented in [DPSZ12] with a protocol for generating smudging values. For a more efficient online-phase these values can be prepared during preprocessing. Therefore we add a function to generate smudging values to the protocol Π_{PREP} ([DPSZ12], Figure 7). Let MPKE be an admissible somewhat homomorphic encryption scheme with parameters as described in Section 4.2.1. Recall that $u \in \mathbb{N}$ is the statistical security parameter. Then the protocol for generating smudging values can be found in Figure 4.7.

With the additional protocol for generating smudging values the players can jointly carry out the decryption securely against an active adversary corrupting up to $m-1$ players. They proceed by calculating the circuit for distributed decryption componentwise using the protocol Π_{ONLINE} given in [DPSZ12], Figure 1. During the initializing step additionally smudging values have to be prepared.

Remark 4.6. In the paper [BDO14] an enhanced version of the SPDZ-protocol is presented which satisfies auditable correctness with security based on the discrete logarithm problem. Since the goal of this work is to base security on alternative assumptions, this approach is not feasible. Similar future results based on a different assumption however could be used with the presented framework to obtain a universal verifiable electronic voting scheme.

5 Public Key Encryption Based on LWE

In this chapter we consider instantiating the electronic voting schemes with public key encryption schemes based on the learning with errors assumption. This allow us to base the privacy of the voting scheme on worst-case lattice problems.

We will pursue two different approaches. First we will present a threshold version of the scheme of Applebaum et al. introduced in [ACPS09]. With this approach distributed decryption is secure against an active adversary corrupting less than a third of the players. Active security is achieved by the error correcting properties of Shamir sharing, thereby making preprocessing unnecessary and avoiding the overhead introduced by saving keys and message authentication codes. On the downside during distributed decryption the authorities have to calculate a linear combination of $\binom{m}{t-1}$ values, therefore this approach is only feasible for a small number of authorities.

As an alternative we show how to customize Regev's scheme and employ the protocols for multi-party computation presented in Chapter 4. The encryption scheme was introduced by O. Regev in [Reg05], equipped with a protocol for threshold decryption in [BD10] and presented with a enlarged message space in [BDOZ11].

As many ideas and constructions already came up in other parts of the work we will just give a very brief overview of the second approach. By Section 2.1 of [BDOZ11] the extension of Regev's scheme is a semi-homomorphic encryption scheme. We thus begin this chapter by showing how to generally obtain a pre-proof of validity for an arbitrary semi-homomorphic cryptosystem (see Definition 4.2).

Lemma 5.1. *Let* $\mathrm{PKE} = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *be a semi-homomorphic cryptosystem. Let* $p \in \mathbb{N}$ *and* $(pk, sk, M, R, D_\sigma^d, G) \leftarrow \mathrm{Gen}(1^\kappa, p)$ *as in Definition 4.2. Let* $B := 2^{u+1} \cdot \sigma$*. Then the protocol* $\Pi_{\text{SH-PPoV}}$ *in Figure 5.1 is a* Σ'*-protocol for the* (B_r, B)*-pre-proof of validity.*

Proof. First we show *completeness.* Recall that for semi-homomorphic cryptosystems the space of randomness is a subspace of $\mathbb{Z}^d$. Let $((pk, v, c), r) \in R_{\mathrm{PPoV}}$. For an honest prover $\|r\|_\infty \leq \sigma$ holds. As every component of r' is chosen from an interval exponentially larger than σ by Lemma 1.8 we have with overwhelming probability $\|r + r'\|_\infty \leq 2^u \cdot \sigma$. Furthermore for $b = 0$ and thus $e = r'$ we get $\mathrm{Enc}_{pk}(0, e) = \mathrm{Enc}_{pk}(0, r') + 0 \cdot \mathrm{Enc}_{pk}(v, r)$ and for $b = 1$ we have $\mathrm{Enc}_{pk}(v, e) = \mathrm{Enc}_{pk}(0, r') + 1 \cdot \mathrm{Enc}_{pk}(v, r)$ by the additive homomorphic properties of the cryptosystem.

Now we take a look at the *special honest verifier computational zero-knowledge.* Let $b \in \{0, 1\}$ and $x = (pk, v, c)$. For $b = 0$ we can simply choose $e \leftarrow [-2^u \cdot \sigma, 2^u \cdot \sigma]^d$ and set $a := \mathrm{Enc}_{pk}(0, e)$. This conversation is distributed identically to the conversation of an honest prover with the verifier. For $b = 1$ we also choose $e \leftarrow [-2^u \cdot \sigma, 2^u \cdot \sigma]^d$ and further $a := \mathrm{Enc}_{pk}(v, e) - c$. By Lemma 1.8 the value e is statistically indistinguishable to the value $e + r$ for any r with infinity norm bounded by σ and therefore the value

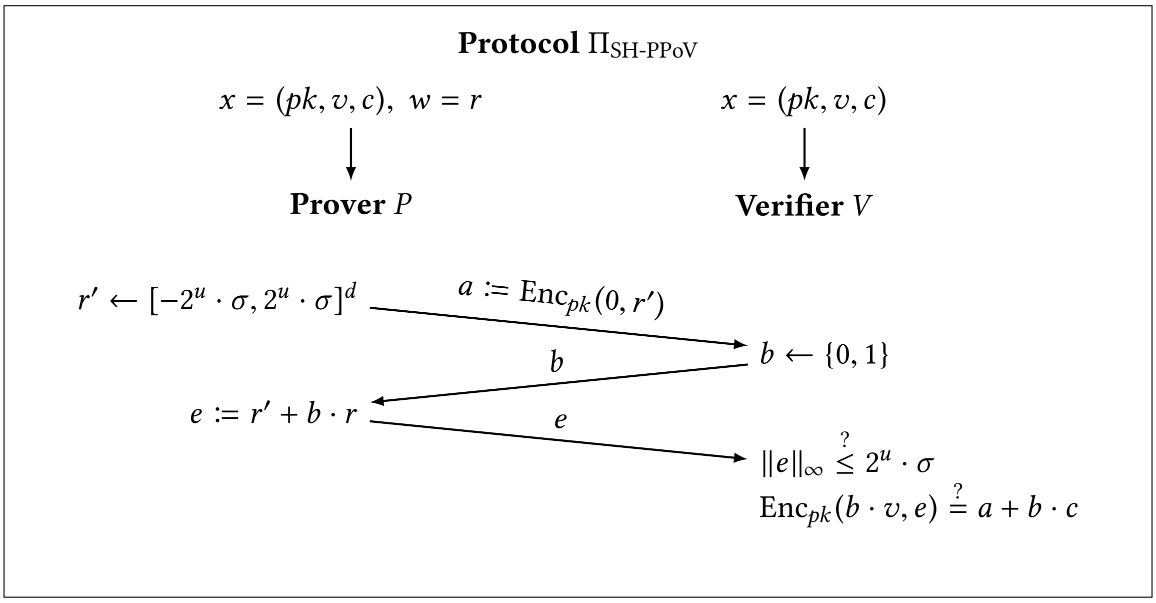
Protocol $\Pi_{\text{SH-PPoV}}$
$x = (pk, v, c),\ w = r$
$x = (pk, v, c)$
Prover P
Verifier V
$r' \leftarrow [-2^u \cdot \sigma, 2^u \cdot \sigma]^d$
$a := \mathrm{Enc}_{pk}(0, r')$
$b \leftarrow \{0, 1\}$
b
$e := r' + b \cdot r$
e
$\|e\|_\infty \overset{?}{\leq} 2^u \cdot \sigma$
$\mathrm{Enc}_{pk}(b \cdot v, e) \overset{?}{=} a + b \cdot c$

Figure 5.1: Protocol for the Pre-Proof of Validity (compare [BDOZ11], Figure 1)

is statistically indistinguishable to the one generated during an honest execution of the protocol. The value a is derived deterministically from the given values and e, as the corresponding equation checked by the verifier is always satisfied for an honest prover, thus a is indistinguishable to the value generated during a real execution as well.

It remains to prove the *special soundness*. Given two accepting conversations $(a, 0, e)$ and $(a, 1, e')$ with $r := e' - e$ it holds $\|r\|_\infty \leq \|e'\|_\infty + \|e\|_\infty \leq 2 \cdot 2^u \cdot \sigma$. Finally this provides $\mathrm{Enc}_{pk}(v, r) = \mathrm{Enc}_{pk}(v, e') - \mathrm{Enc}_{pk}(0, e) = (a + c) - a = c$.

□

5.1 ACPS Encryption Scheme

In this section we present a cryptosystem adapted from [ACPS09] based on the learning with errors assumption. Recall that in contrast to ring learning with errors a reduction from the search to the decision variant of the learning with errors problem is known even for certain exponential moduli (compare Section 1.2). This allows us to base the security directly on worst-case lattice problems. On the downside the ciphertexts lie in a vector space over a ring instead of a field and therefore the protocols presented in Chapter 4 cannot be used to achieve security against active adversaries during decryption. Instead we will present a method for threshold decryption adapted from [BD10] which achieves security against an active adversary corrupting less than a third of the players.

The following Definition is from [ACPS09] with a slight adaption of the parameters, as for distributed decryption we require the ciphertext space to be of size super-polynomial.

Definition 5.2 ([ACPS09]). Let $p := p(\kappa) \in \mathbb{P}$ be polynomial bounded such that there exists a $\delta := \delta(\kappa) \in \omega(1)$ with $\delta < (p-1)/(8\kappa^5 \log p)$. Let $q := p^\delta \in \mathbb{N}$ be a multiple of p.

The parameter p fixes the message space of our cryptosystem $\mathbb{Z}_p$ and the parameter q the domain of the ciphertexts which will be a vector space over $\mathbb{Z}_q$.

The choice of q is the first change to the original cryptosystem. We need to choose q super-polynomial to ensure that during threshold decryption no information about the secret key is leaked. We will work with $q \in O(2^u)$. This change enforces to adjust the dimension $n := n(\kappa) \in \mathbb{N}$ of the secret key to equal κ^2 (see [Reg09]). The parameter $\alpha := \alpha(\kappa) \in \mathbb{N}$ of the discrete Gaussian distribution $\lambda := \overline{\psi}_\alpha$ used for encryption has to comply with $\alpha \geq n/q$ to invoke the worst-case lattice connections (see [Pei09], [Reg09]) and on the other hand to achieve correctness of decryptions it cannot be too large. We will come back to the upper bound on α later.

The encryption scheme ACPS consists of three polynomial time algorithms Gen, Enc and Dec complying with the following properties.

Key generation: The algorithm Gen takes κ as input chooses a vector

$$\mathbf{s} \leftarrow \lambda^n$$

and returns it as the secret key. Let μ be the smallest natural number such that $\mu \geq 2(n+1)\log q$. Then further Gen draws a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times\mu}$ uniformly random. The restriction on μ is needed to obtain pseudorandomness in the first component of the ciphertexts defined later. For the second part of the public key Gen chooses a vector $\mathbf{e} \in \mathbb{Z}_q^\mu$ according to the distribution λ^μ, sets $\mathbf{b} := \mathbf{A}^\top \mathbf{s} + \mathbf{e}$ and outputs

$$(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times\mu} \times \mathbb{Z}_q^\mu$$

as the public key. Let $i \in [\mu]$ and let $\mathbf{a}_i$ denote the i-th column of $\mathbf{A}$ and b_i the i-th entry of $\mathbf{b}$. Then $(\mathbf{a}_i, b_i)$ is distributed according to $A_{\mathbf{s},\lambda}$.

Additionally we assume a protocol for generation and distribution of the secret key implementing the ideal functionality $\mathcal{F}_{\text{ACPS-KeyGen}}$ (Figure 5.2), for instance the protocol presented in Chapter 6 of [BD10] with adapted parameters can be used here.

Encryption: Let $r, r' \in \mathbb{R}$ be parameters with $r \in \omega\left(\sqrt{\log \mu}\right)$ such that $r \leq \sqrt{\mu}$ and $r' = r \cdot \sqrt{\mu} \cdot \left(\alpha + \frac{1}{2q}\right)$. This choice of parameters will be needed for correctness (see Lemma 5.3 and Lemma 5.5). The algorithm Enc takes a public key $pk = (\mathbf{A}, \mathbf{b})$ and a message $z \in \mathbb{Z}_p$ as input, draws $\mathbf{r} \leftarrow D_{\mathbb{Z}^\mu, r}$ and $e \leftarrow \overline{\psi}_{r'}$ and returns the ciphertext

$$(\mathbf{A}\mathbf{r}, \langle \mathbf{r}, \mathbf{b}\rangle + e + z \cdot p^{\delta-1}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

Decryption: On input of the ciphertext $c := (\mathbf{u}, v) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and secret key $\mathbf{s} \in \mathbb{Z}_q^n$ the algorithm Dec returns the value

$$z \in \mathbb{Z}_p,$$

such that $z \cdot p^{\delta-1}$ is closest to $v - \langle \mathbf{u}, \mathbf{s}\rangle \mod q$.

Functionality $\mathcal{F}_{\text{ACPS-KeyGen}}$

- Upon receiving "*start*" from all honest players, run the algorithm for key generation to obtain the secret key $\mathbf{s} \in \mathbb{Z}_q^n$ and the public key $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times\mu} \times Z_q^{\mu}$. Further for each set $T \subseteq [m]$ of size exactly $t-1$ choose a key $K_T \leftarrow \mathbb{Z}_q$ uniformly random.
- For each $j \in I_C$ receive a share $(s_{j,k})_{k\in[n]} \in \mathbb{Z}_q^n$ from the adversary. Calculate for each honest player P_i shares $(s_{i,k})_{k\in[n]} \in \mathbb{Z}_q^n$ such that for each $k \in [n]$ the values $(s_{i,k})_{i\in[m]}$ form a (t, m)-Shamir sharing of the k-th component of the secret key. Send the public key pk to each player and further send to each player P_i privately the vector of shares $(s_{i,k})_{k\in[n]}$ and the key K_T if and only if $i \notin T$.

Figure 5.2: Ideal Functionality for Distributed Key Generation (see [BD10], Section 4.1)

Now we can establish the upper bound on α. For correctness amongst others we need

$$\alpha \leq \frac{p-1}{4 \cdot q \cdot \kappa \cdot r \cdot \sqrt{\mu}}.$$

If p is large enough the careful choice of δ ensures that there exist instantiations of parameters such that additionally $n \leq (p-1)/(4 \cdot \kappa \cdot r \cdot \sqrt{\mu})$. This ensures that at the same time the requirement $\alpha \geq n/q$ can be met.

5.1.1 Security and Proof of Validity

We will begin this section by considering security of the encryption scheme and proceed to provide the protocol for the pre-proof of validity. All parameters we do not specify explicitly are assumed to be as in Definition 5.2.

The following Lemma is taken from [ACPS09]. It shows that the distribution of ciphertexts is statistically indistinguishable to the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ if the $\text{DLWE}_{q,\overline{\psi}_\beta}$ assumption holds.

Lemma 5.3 ([ACPS09] Lemma 4, Security)**.** *Let all parameters be chosen as in Definition 5.2. Then there exists a real number $\beta \leq \sqrt{2}r'$ such that the distribution of encryptions of $0 \in \mathbb{Z}_p$ is within negligible statistical distance of $A_{\mathbf{s},\overline{\psi}_\beta}$, where the probability is taken over the choice of the public key.*

The goal of [ACPS09] was to construct a KDM-secure cryptosystem. We do not need this property for our purposes, but want to state their main result.

Theorem 5.4 ([ACPS09], Theorem 2, KDM-Security)**.** *Let q and λ be defined as before. Assuming $\text{LWE}_{q,\lambda}$ is hard, the described cryptosystem is KDM-secure with respect to the family $\mathcal{F}$ of affine functions.*

The following Lemma serves two purposes. First it shows that Lemma 1.12 is applicable. Together with Lemma 5.3 and the explanations provided in Section 1.2 this allows to reduce the security of ACPS to the worst-case lattice problem GapSVP. Secondly we will use it to prove additive homomorphic properties of ACPS in the following section.

Lemma 5.5 (Compare [ACPS09], Lemma 6)**.** *Let all parameters be chosen as in Definition 5.2. Then for the noise term* $\tilde{e} := \langle \mathbf{r}, \mathbf{e} \rangle + e$ *in ciphertexts there exists a function* $\mathrm{negl}\colon \mathbb{N} \to \mathbb{R}_{>0}$ *which is negligible and for which we have*

$$\Pr\left[|\tilde{e}| > \frac{p-1}{2}\right] \le \mathrm{negl}(\kappa).$$

Proof. By Lemma 5.3 there exists a $\beta \in \mathbb{R}$ with $\beta \le \sqrt{2}r'$ such that $\tilde{e}$ is drawn from a distribution statistically indistinguishable to $\overline{\psi}_\beta$. Therefore we have for all $\kappa > 2$

$$\beta \le \sqrt{2}r' \le \sqrt{2} \cdot r \cdot \sqrt{\mu} \cdot \left(\alpha + \frac{\alpha}{2n}\right) \le 2 \cdot r \cdot \sqrt{\mu} \cdot \alpha \le \frac{p-1}{2 \cdot q \cdot \kappa},$$

where the last inequality holds by the upper bound on α established before. By applying Lemma 1.4 with $t = \kappa$ we obtain for all $\kappa > \kappa_0$

$$\begin{aligned}\Pr\left[|\tilde{e}| > \frac{p-1}{2}\right] &\le \Pr\left[|\tilde{e}| > \kappa \frac{q \cdot (p-1)}{\sqrt{2\pi} \cdot 2 \cdot q \cdot \kappa}\right] \\ &\le \Pr\left[|\tilde{e}| > \kappa \frac{q \cdot \beta}{\sqrt{2\pi}}\right] \le 2e^{\frac{-\kappa^2}{2}},\end{aligned}$$

which is negligible in in κ. □

Next we want to give the protocol for the pre-proof of validity. More precise let $B'_r := B'_r(\kappa) \in \mathbb{N}$ be such that p/B'_r is negligible in κ and such that $B'_r < 1/2 \cdot p^{\delta-1}$ for all $\kappa \in \mathbb{N}$. Then we give a Σ'-protocol for the relations

$$\begin{aligned}R_{\text{ACPS-PPoV}} := \{(x, w) \mid x = (pk, c, z) \in \mathbb{Z}_q^{(n\times\mu)+\mu} \times \mathbb{Z}_q^{n+1} \times \mathbb{Z}_p, w = (\mathbf{r}, e) \in \mathbb{Z}_q^{\mu} \times \mathbb{Z}_q,\\ |\mathbf{r}|_\infty \le p, |\mathbf{e}| \le p \text{ and } c = \mathrm{Enc}_{pk}(z; \mathbf{r}, e)\}\end{aligned}$$

and

$$\begin{aligned}R'_{\text{ACPS-PPoV}} := \{(x, w) \mid x = (pk, c, z) \in \mathbb{Z}_q^{(n\times\mu)+\mu} \times \mathbb{Z}_q^{n+1} \times \mathbb{Z}_p, w = (\mathbf{r}, e) \in \mathbb{Z}_q^{\mu} \times \mathbb{Z}_q,\\ |\mathbf{r}|_\infty \le B'_r, |\mathbf{e}| \le B'_r \text{ and } c = \mathrm{Enc}_{pk}(z; \mathbf{r}, e)\}.\end{aligned}$$

Note that if the randomness for generating a ciphertext is drawn honestly we have $|\mathbf{r}|_\infty \le p$ and $|\mathbf{e}| \le p$ by Lemma 5.5. From now on we will only allow randomness which complies with this restriction to be used for encryptions. In case it is not satisfied, players are directed to sample fresh randomness. As mentioned this only happens with negligible probability.

The Σ'-protocol is based on the proof of plaintext knowledge given in [BDOZ11]. Let $I := [-B'_r, B'_r] \cap \mathbb{Z}_q$ and let further pk be the public key known by both participating parties. Further addition and multiplication are meant to be componentwise. The protocol for the pre-proof of validity is given in 5.3. We omit the proof of the following Lemma as it works analogously to the proof of Lemma 5.1.

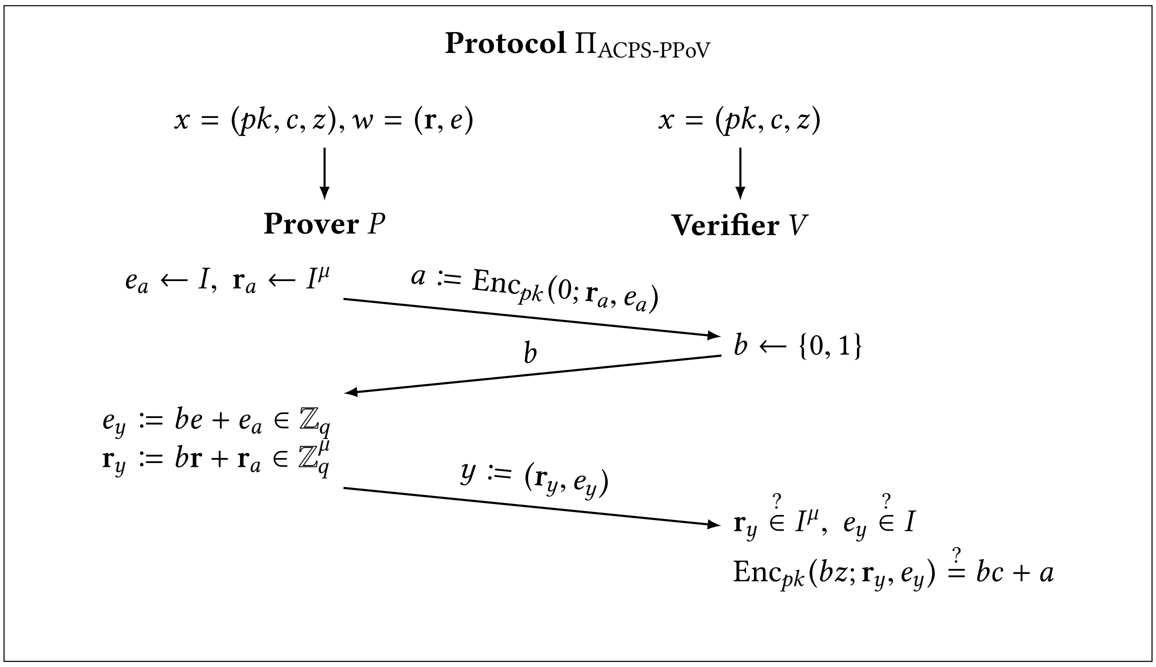
Protocol $\Pi_{\text{ACPS-PPoV}}$
$x = (pk, c, z), w = (\mathbf{r}, e)$
$x = (pk, c, z)$
Prover P
Verifier V
$e_a \leftarrow I,\ \mathbf{r}_a \leftarrow I^\mu$
$a := \text{Enc}_{pk}(0; \mathbf{r}_a, e_a)$
$b \leftarrow \{0,1\}$
b
$e_y := be + e_a \in \mathbb{Z}_q$
$\mathbf{r}_y := b\mathbf{r} + \mathbf{r}_a \in \mathbb{Z}_q^\mu$
$y := (\mathbf{r}_y, e_y)$
$\mathbf{r}_y \overset{?}{\in} I^\mu,\ e_y \overset{?}{\in} I$
$\text{Enc}_{pk}(bz; \mathbf{r}_y, e_y) \overset{?}{=} bc + a$

Figure 5.3: Protocol for the Pre-Proof of Validity of ACPS (compare [BDOZ11], Figure 1)

Lemma 5.6. *The protocol $\Pi_{\text{ACPS-PPoV}}$ presented in Figure 5.3 is a Σ'-protocol for the relations $R_{\text{ACPS-PPoV}} \subseteq R'_{\text{ACPS-PPoV}}$.*

5.1.2 Distributed Decryption

In this section we first consider additive homomorphic properties of ACPS and then give a protocol for distributed decryption adapted from [BD10].

Lemma 5.7 (Additive Homomorphic Properties)**.** *Let all parameters be defined as in Definition 5.2. Let $B'_r := B'_r(\kappa) \in \mathbb{N}$ such that p/B'_r is negligible and $B_\epsilon := B_\epsilon(\kappa) \in \mathbb{N}$. Let $B := B(\kappa) \in \mathbb{N}$ such that $B \cdot B'_r + B_\epsilon < 1/2 \cdot p^{\delta-1}$. Note that later we will need $B_\epsilon \geq \binom{m}{t-1} \cdot 2^u \cdot B \cdot B'_r$. For each $i \in [B_r]$ let a ciphertext $c_i := \text{Enc}_{pk}(z_i; \mathbf{r}_i, e_i)$ be given such that the error term $\tilde{e}_i := \langle \mathbf{r}_i, \mathbf{e} \rangle + e_i$ is bounded by B'_r with overwhelming probability. Then for all ϵ with $|\epsilon| \leq B_\epsilon$ we have*

$$\text{Dec}_{sk}\left(\sum_{i=1}^{B_r} c_i + (0, \epsilon)\right) = \sum_{i=1}^{B_r} z_i$$

with overwhelming probability.

Functionality $\mathcal{F}_{\text{ACPS-Dec}}$

- Upon receiving "*start*" from all honest players, run the algorithm for key generation to obtain the secret key $\mathbf{s} \in \mathbb{Z}_q^n$ and the public key $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times\mu} \times Z_q^{\mu}$, send pk to the adversary and store $\mathbf{s}$.
- Upon receiving "*decrypt c*" from at least $m - t + 1$ players send c and $v := \mathrm{Dec}_{\mathbf{s}}(c)$ to all players.

Figure 5.4: Ideal Functionality for Distributed Decryption

Proof. We have

$$\begin{aligned}\sum_{i=1}^{B_r} c_i &= \left(\sum_{i=1}^{B_r} \mathbf{A}\mathbf{r}_i, \sum_{i=1}^{B_r} \left(\langle \mathbf{r}_i, \mathbf{b}\rangle + e_i + z_i \cdot p^{\delta-1}\right)\right)\\ &= \left(\sum_{i=1}^{B_r} \mathbf{A}\mathbf{r}_i, \sum_{i=1}^{B_r} \left(\left\langle \mathbf{r}_i, \mathbf{A}^\top \mathbf{s}\right\rangle + \langle \mathbf{r}_i, \mathbf{e}\rangle + e_i + z_i \cdot p^{\delta-1}\right)\right)\\ &= \left(\mathbf{A}\sum_{i=1}^{B_r} \mathbf{r}_i, \left\langle \sum_{i=1}^{B_r} \mathbf{r}_i, \mathbf{A}^\top \mathbf{s}\right\rangle + \sum_{i=1}^{B_r} \tilde{e}_i + \left(\sum_{i=1}^{B_r} z_i\right) \cdot p^{\delta-1}\right).\end{aligned}$$

By the triangle inequality and the preconditions on the parameters we have

$$\left|\sum_{i=1}^{B_r} \tilde{e}_i\right| \leq B_r \cdot B_r' < \frac{1}{2} \cdot p^{\delta-1} - B_\epsilon$$

with overwhelming probability and thus rounding to the closest multiple of $p^{\delta-1}$ during the last step of decryption yields $\sum_{i=1}^{B_r} z_i$.

□

To obtain a protocol for distributed decryption we use the methods presented in [BD10]. Let $B_{r_2} := 2^u \cdot B \cdot B_{r_1}$ and let $\phi\colon \mathbb{Z}_q \times (\mathbb{Z}_q^n \times \mathbb{Z}_q) \to [-B_{r_2}, B_{r_2}]$ be a pseudo-random function. The protocol for distributed decryption can be found in Figure 5.5. The following theorem describes the security of the protocol for distributed decryption given in Figure 5.5 is secure against a static active adversary corrupting up to $\lceil m/3 - 1\rceil$ players. The ideal functionality given in 5.4 captures that the adversary is not allowed to influence the result of the decryption.

Theorem 5.8 (Compare [BD10], Theorem 3)**.** *Let $m \in \mathbb{N}$ be the number of decrypting authorities and $t \in \mathbb{N}$ such that $t - 1 < m/3$. Let $B_r' := B_r'(\kappa) \in \mathbb{N}$ such that the error term of all ciphertexts to be decrypted is bounded by B_r' with overwhelming probability. Let $B_{r_2} := B_{r_2}(\kappa) \in \mathbb{N}$ be such that B_r'/B_{r_2} is negligible in κ, for instance $B_{r_2} := 2^u \cdot B_r'$, and*

$B'_r + \binom{m}{t-1} \cdot B_{r_2} < 1/2 \cdot p^{\delta-1}$ and let further $\phi\colon \mathbb{Z}_q \times (\mathbb{Z}_q^n \times \mathbb{Z}_q) \to [-B_{r_2}, B_{r_2}]$ be a pseudo-random function. Then the protocol $\Pi_{\text{ACPS-Dec}}$ (Figure 5.5) implements the ideal functionality $\mathcal{F}_{\text{ACPS-Dec}}$ (Figure 5.4) in the $\mathcal{F}_{\text{ACPS-KeyGen}}$-hybrid model (Figure 5.2) with statistical security against any static active adversary corrupting up to $t-1$ parties.

Proof. First we show that the real-world protocol yields the correct plaintext with overwhelming probability. We claim that the recovered Shamir shared value equals $f + x$, where $f := v - \langle \mathbf{u}, \mathbf{s}\rangle$ and $x := \sum_{T \subseteq [m], |T| = t-1} \phi(K_T, c)$. Note that the corrupted players corrupted sending wrong values do not have an impact on the recovered value. This follows by Remark 1.26 as $t - 1 < m/3$. In the following we can thus assume the values sent were correct.

The values $(f_j)_{j \in [m]}$ form a Shamir sharing of f by the linearity of Shamir sharing. Let $g := \sum_{T \subseteq [m], |T| = t-1} \phi(K_T, c) \cdot g_T$. For all $i \in [m]$ it holds $g(i) = x_i$ as $g_T(j) = 0$ for all $j \in T$. As further g has degree at most $t - 1$, the values $(x_j)_{j \in [m]}$ form a Shamir sharing of $g(0) = \sum_{T \subseteq [m], |T| = t-1} \phi(K_T, c) = x$, which proves the first claim. Since the image of the function ϕ is $[-B_{r_2}, B_{r_2}]$, the absolute value of x is bounded by $\binom{m}{t-1} \cdot B_{r_2}$. Now by Lemma 5.7 it suffices to show $B'_r + \binom{m}{t-1} \cdot B_{r_2} < 1/2 \cdot p^{\delta-1}$, which holds by assumption. It remains to give a simulator $\mathcal{S}_{\text{ACPS-Dec}}$ working on top of the ideal functionality $\mathcal{F}_{\text{ACPS-Dec}}$ simulating the protocol $\Pi_{\text{ACPS-Dec}}$.

Simulator $\mathcal{S}_{\text{ACPS-Dec}}$:

Key Generation: Upon receiving "*start*" the simulator forwards the command to the ideal functionality $\mathcal{F}_{\text{ACPS-KeyGen}}$ and obtains the public key $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n \times \mu} \times \mathbb{Z}_q^{\mu}$. The simulator sends the public key to all players and further receives for each $j \in I_C$ a share $(s_{j,k})_{k \in [n]}$ from the adversary. Now the simulator calculates for each $i \in I_C$ a vector of shares $(s_{i,k})_{k \in [n]}$, such that for each $k \in [n]$ the values $(s_{j,k})_{j \in [m]}$ form a (t, m)-Shamir sharing of $0 \in \mathbb{Z}_q^n$. Further for each $j \in I_C$ he chooses for each set $T \subset [m]$ of size exactly $t - 1$ a key $K_T \leftarrow \mathbb{Z}_q$ uniformly random. For each $j \in I_C$ such that $j \notin T$ he sends K_T to player P_j.

Decryption: When receiving "*decrypt c*" the simulator forwards the command to the ideal functionality $\mathcal{F}_{\text{ACPS-KeyGen}}$ and obtains $z = \mathrm{Dec}_{\mathbf{s}}(c)$. For each $j \in I_C$ the simulator can calculate the value ω_j with the information from the first step. If further $|I_C| < t - 1$ the simulator takes a set $I \subseteq I_H$ of size $t - 1 - |I_C|$ and calculates ω_i for all $i \in I$ accordingly. Let

$$K := \{T \subseteq [m] \mid |T| = t - 1, \exists j \in I_C \cup I \colon j \notin T\}$$

and

$$U := \{T \subseteq [m] \mid |T| = t - 1, T \notin K\}.$$

As $A \cap I$ has size $t - 1$ and is not in K, the set U is not empty. Now the simulator draws for each $T \in U$ a value $y_T \leftarrow [-B_{r_2}, B_{r_2}]$ uniformly random and defines $y := \sum_{T \in K} \phi(K_t, c) + \sum_{T \in U} y_T$. Next he calculates the distinct polynomial f of degree at most $t - 1$ such that $f(j) = \omega_j$ for all $j \in I_C \cup I$ and $f(0) = y + z \cdot p^{\delta-1}$. Finally the simulator defines $\omega_i := f(i)$ for all $i \in [m] \backslash (I_C \cup I)$ and sends all the shares to the adversary.

Protocol $\Pi_{\text{ACPS-Dec}}$

Initialization: Let $c = (\mathbf{u}, v) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{u} = (u_i)_{i=1}^n$ the ciphertext to be decrypted. For each $j \in [m]$ player P_j calculates $f_j := v - \sum_{i=1}^n u_i \cdot s_{i,j}$. Let for all $T \subseteq [m]$ the function g_T be the unique polynomial of degree $t-1$ such that $g_T(0) = 1$ and $g_T(j) = 0$ for all $j \in T$. Then each player P_j further calculates $x_j := \sum_{T \subseteq [m], |T|=t-1, j \notin T} g_T(j) \cdot \phi(K_T, c)$ and sets $\omega_j := f_j + x_j$.

Decryption: For each $j \in [m]$ the player P_j broadcasts ω_j and each player computes the encrypted message by recovering the (t, m)-Shamir shared value possibly using error correction (compare Remark 1.26). Finally each player can recover the plaintext by computing $z \in \mathbb{Z}_p$ such that $z \cdot p^{\delta-1}$ is closest to the obtained value.

Figure 5.5: Protocol for Threshold Decryption (compare [BD10], Section 4.2)

It remains to show that an arbitrary environment $\mathcal{Z}$ cannot decide except with negligible probability whether it is interacting with the real-world protocol or the simulator on top of the ideal functionality. The keys for the pseudo-random function are generated identically and therefore indistinguishable to the ones in real-world protocol. The same holds for the shares ω_j for all $j \in I_C$, because they are computed deterministically with the information received from the adversary and the keys generated during the first step. By the security of Shamir sharing the values ω_i for all $i \in I$ are indistinguishable to the shares generated during an execution of the real-world protocol as the total shares of the secret key known by the adversary and involved in the calculations are $t-1$.

Now it suffices to show that $f + x$ computed according to the real-world protocol is indistinguishable to $z \cdot p^{\delta-1} + y$ computed by the simulator, as all other shares are then derived deterministically. First note that x and y are indistinguishable, as the keys K_T for all $T \in U$ are not known by the adversary and ϕ is assumed to be pseudo-random. Thus it remains to show that $f + y$ and $z \cdot p^{\delta-1} + y$ are indistinguishable.

Let $\tilde{e}$ be the noise term of c as defined in Lemma 5.5, then we have $f + y = \tilde{e} + z \cdot p^{\delta-1} + y$. By assumption $\tilde{e}$ is with overwhelming probability bounded by B'_r and further B'_r / B_{r_2} is negligible in κ. Let y_r be a component of y drawn uniformly random from $[-B_{r_2}, B_{r_2}]$, such a component always exists as U is not empty. Then by Lemma 1.8 the distribution of $y_r + \tilde{e}$ is statistically indistinguishable to the distribution of y_r, which proves the claim.

□

5.2 Regev's Encryption Scheme

In the previous section we showed how to obtain an electronic voting scheme with hardness based on the worst-case lattice problem GapSVP. But at the same time we would like to use the protocols for multi-party computation presented by Bendlin et al. in [BDOZ11].

One way to accomplish this is by choosing a public key encryption scheme with ciphertext space isomorphic to the direct product of at most polynomially many finite fields.

This is the case for the cryptosystem first introduced by Regev in [Reg05]. A threshold version of this cryptosystem with a new choice of parameters was presented in [BD10]. We present an alternative threshold version based on the results of [BDOZ11]. Using their techniques we obtain a protocol for threshold decryption secure against active adversaries corrupting less than half the players. Note that the protocol of [DPSZ12] could be used instead, but as the security of the preprocessing phase of this protocol is based on learning with errors over rings with exponential moduli, the cryptosystems based on learning with errors over rings presented in Chapter 3 provide a more efficient solution.

We first give the definition of an extended version of Regev's Scheme (see [Reg05], [BD10] and [BDOZ11]).

Definition 5.9 (Regev's Scheme)**.** Let $e := e(\kappa) \in \mathbb{N}$ be polynomially bounded and for $i \in [e]$ let $p_i := p_i(\kappa) \in \mathbb{P}$ be polynomially bounded such that $p_i > m$ and for all $i, j \in [e]$ with $i \neq j$ it holds $p_i \neq p_j$. Let $q := q(\kappa) := \prod_{i=1}^{e} p_i(\kappa) \in \mathbb{N}$. Let $n := n(\kappa) := \kappa \in \mathbb{N}$ and $\mu := \mu(\kappa) \in \mathbb{N}$ such that $\mu \in O(n^3)$. Further let $\alpha := \sqrt{2\pi} q^{\beta-1}$ for a $\beta \in (0, 1)$. Recall that than $\overline{\psi}_\alpha$ is the discrete Gaussian distribution on $\mathbb{Z}_q$ with standard deviation q^β. Finally let $p := p(\kappa) \in \mathbb{N}$ be polynomially bounded and such that for all $k \in \mathbb{N}$ it holds $p(k) < 1/4\sqrt{q(k)}$.

Regev's scheme consists of a tuple of polynomial time algorithms Gen, Enc and Dec) complying with the following properties.

Key generation: The algorithm Gen for key generation draws a vector

$$\mathbf{s} \leftarrow \mathbb{Z}_q^n$$

as secret key. Then it chooses a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{\mu \times n}$ uniformly random and a vector $\mathbf{e} \in \mathbb{Z}_q^\mu$ by drawing each component of $\mathbf{e}$ according to the distribution $\overline{\psi}_\alpha$. It defines $\mathbf{b} := \mathbf{A}^\top \mathbf{s} + \mathbf{e}$ and returns the public key

$$(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{\mu \times n} \times \mathbb{Z}_q^\mu.$$

The protocol for distributed key generation can be found in Chapter 6 of [BD10]. Note that for each $i \in \{t, \cdots m\}$ a secret key $\mathbf{s}^{(i)}$ has to be chosen and (i, m)-Shamir shared between the authorities for employing the protocol presented in Section 4.1.

Encryption: The algorithm Enc takes a message $z \in \mathbb{Z}_p$ and a public key $pk = (\mathbf{A}, \mathbf{b})$ as input, draws $\mathbf{r} \leftarrow \{-1, 0, 1\}^\mu$ uniformly random and outputs the ciphertext

$$(\mathbf{A}^\top \mathbf{r}, \langle \mathbf{r}, \mathbf{b} \rangle + e + z \cdot \lfloor \frac{q}{p} \rfloor) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

Decryption: On input of a ciphertext $c := (\mathbf{u}, v) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and a secret key $\mathbf{s} \in \mathbb{Z}_q^n$ the algorithm Dec returns

$$z \in \mathbb{Z}_p,$$

Protocol $\Pi_{\text{Regev-Dec}}$

Initialization: Let $I \subseteq [m]$ be the index set of players wanting to decrypt a ciphertext $c = (\mathbf{u}, v) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{u} = (u_i)_{i=1}^n$. Let $f := v - \langle \mathbf{u}, \mathbf{s}^{|I|} \rangle \mod q$. For each $j \in I$ the player P_j calculates his share $f_j = v - \sum_{i=1}^n u_i \cdot s_{i,j}^{|I|}$ of f. Then P_j chooses an integer $x_j \leftarrow \left[-\sqrt{q}, \sqrt{q}\right]$ uniformly random and sets $\sigma_j := f_j + d_{j,I} \cdot x_j$.

Decryption: For each $i \in I$ the player P_i broadcasts σ_j to the other players. Let $x := \sum_{j \in I} x_j$. Given the shares of all players every player can recover the $(|I|, m)$-Shamir shared value $\sigma := f + x$ and obtain the plaintext by calculating $z \in \mathbb{Z}_p$ such that $z \cdot \lfloor q/p \rfloor$ is closest to σ.

Figure 5.6: Protocol for Threshold Decryption (compare [BD10], Section 4.2 and [DPSZ12], Figure 8)

such that $z \cdot \lfloor q/p \rfloor$ is closest to $v - \langle \mathbf{u}, \mathbf{s} \rangle \mod q$.

In Figure 5.6 we give the m-party protocol for secure description, which can be used together with the multi-party computation protocol of [BDOZ11]. We omit the proof here, as the simulator is similar to the one given in the proof of Theorem 3.6.

To get security and correctness we cite two theorems. The proofs can be found in [BD10]. For additive homomorphic properties of the extended version of Regev's scheme we refer to [BDOZ11].

Theorem 5.10 ([BD10] Theorem 1 and Lemma 1, Correctness). *The algorithm* Dec *returns the correct output with overwhelming probability.*

Theorem 5.11 ([BD10] Theorem 2, Security). *Regev's scheme is secure assuming* GapSVP *is hard in the worst case.*

As conclusion of this chapter we give an idea on how reuse the protocols customized in Chapter 4 to rings which are isomorphic to the direct product of finite fields.

The ring of ciphertexts of Regev's scheme $\mathbb{Z}_q$ is isomorphic to $\prod_{i=1}^e \mathbb{Z}_{p_i}$ by the Chinese remainder theorem. Let $p \in \mathbb{P}$ an arbitrary prime. The protocol for obtaining the message authentication codes and associated keys in the Shamir sharing setting for the field $\mathbb{F}_p$ can be found in section 4.1. We set $N = 1$, as then we have $\phi_N = X - 1$ and $\mathbb{F}_p[X]/\phi_1 \cong \mathbb{F}_p$. The idea now is to perform the preprocessing for each of the respective fields $\mathbb{F}_{p_i}$ and then use the isomorphism to obtain the keys and codes for message authentication over $\mathbb{F}_q$. Finally the arithmetic circuit can be calculated over $\mathbb{F}_q$ securely against a static active adversary corrupting less than half of the players. As in our setting p_i is polynomially bounded, for each component several message authentication codes have to be generated.

6 Efficiency

To conclude this work we take a closer look at the efficiency of the electronic voting scheme. We provide an overview of the original version compared to the established instantiations in Table 6.1. Recall that we are working with two security parameters κ and u, where the latter can be chosen much smaller in practice, as it does not depend on the computational power of the adversary. We account for a modular multiplication of two numbers with complexity quadratic in the size of the factors. The first row comprises the efficiency of the original voting scheme, which can be found in [CGS97].

For the instantiations presented in the scope of this work we start by considering the work on the voters side. Each voter has to encrypt his vote and publish an encryption accompanied by a proof of validity, where in all presented instantiations the complexity of the latter is the complexity for performing an encryption multiplied by the number of iterations necessary to obtain negligible soundness error.

The most efficient approach is NTRU, where encrypting is in $\mathcal{O}((u\kappa)^2)$. As the challenge space of the proof of validity of [BCK+14] is of size $\mathcal{O}(\kappa)$, the number of necessary iterations to obtain soundness error in $\mathcal{O}(2^{-u})$ is $\lceil u/\log\kappa\rceil$. Altogether this yields a complexity of $\mathcal{O}(\lceil u/\log\kappa\rceil(u\kappa)^2)$, which for choice $u \leq \sqrt[3]{\kappa}$ is asymptotically better than the original version, where encryption requires exponentiation.

For the cryptosystem ACPS by Applebaum et al. the stronger security assumption is accompanied by a major trade-off in efficiency. The reduction to the worst-case lattice problem GapSVP with super-polynomial modulus enforces the dimension of the secret key to be quadratic in κ. The encryption is dominated by a matrix-vector-multiplication, which has complexity $\mathcal{O}(u^2\kappa^4)$. As furthermore the ciphertext space of the proof of validity consists of solely two elements, the total work on the voters side is $\mathcal{O}(u^3\kappa^4)$.

Following the approach of [BD10] the ciphertext space of Regev's scheme has size exponential in κ and public key of size $\mathcal{O}(\kappa^4)$, resulting in complexity $\mathcal{O}(\kappa^5)$ for encrypting a vote and complexity $\mathcal{O}(u\kappa^5)$ for generating the accompanied proof of validity.

We proceed with examining the second phase. First the authorities have to check the proofs of validity and accept or discard the corresponding votes respectively. Let $|C_{\text{Enc}}|$ be the complexity of the encryption circuit of the respective cryptosystems. Recall that $|C_{\text{Enc}}| \in \mathcal{O}((u\kappa)^2)$ for NTRU, $|C_{\text{Enc}}| \in \mathcal{O}(u^2\kappa^4)$ for ACPS and $|C_{\text{Enc}}| \in \mathcal{O}(\kappa^5)$ for Regev's Scheme. Then in all considered cases checking one vote has complexity in $\mathcal{O}(|C_{\text{Enc}}|)$, resulting in a total complexity $\mathcal{O}(M|C_{\text{Enc}}|)$ of the validation step.

The last step to consider is joint decryption. For NTRU we first consider the passively secure variant and then compute the overhead introduced by the protocols for achieving security against active adversaries. The running time of decryption is dominated by a multiplication and thus of complexity in $\mathcal{O}((u\kappa)^2)$.

Using the multi-party protocol introduced in [BDOZ11] the message authentication introduces an overhead of $\mathcal{O}(m^2)$ and a factor t as the protocol has to be repeated t times

Cryptosystem	Problem	Casting Votes	Tallying*
ElGamal	DDH	$O\left(\kappa^3\right)$	$O\left(M\kappa^3 + m\kappa^3\right)$
NTRU (SPDZ)	RLWE	$O\left(\lceil u/\log\kappa\rceil (u\kappa)^2\right)$	$O\left(M(u\kappa)^2 + m(u\kappa)^2 + m^3 u\kappa\right)^*$
ACPS	GapSVP	$O\left(u^3\kappa^4\right)$	$O\left(Mu^2\kappa^4 + \binom{m}{t-1}u\right)$
Regev (BDOZ)	GapSVP	$O\left(u\kappa^5\right)$	$O\left(M\kappa^5 + m^2 tu\kappa\right)^*$

Table 6.1: Efficiency measures for an (M, m, t)-electronic voting scheme implemented with different cryptosystems. In practice u can be chosen such that $u \ll \kappa$. *Tallying is considered without preprocessing for NTRU and Regev's Scheme.

in the worst-case. The SPDZ-protocol gets along with a single global key instead of a seperate key for each pair of parties and therefore solely introduces an overhead linear in the number of authorities. Additionally $O(m^3)$ elementary operations over the underlying ring have to be performed during the input and output phase of the SPDZ-protocol, we refer to [DPSZ12] for more details. In most applications we have $M \gg m$ and thus the dominating part of tallying is the validation of votes.

In both protocols the online phase is proceeded by a preprocessing phase. As this phase can be executed before the election and depends on the respective underlying cryptosystem MPKE, we do not list it in the overview provided in Table 6.1. We will come back to the complexity of the preprocessing phase after discussing the complexity of the joint decryption of the remaining cryptosystems.

For ACPS decryption is dominated by calculating the respective shares of the smudging value, which consists of $\binom{m}{t-1}$ summands of bit size in $O(u)$. Recall that we require $t < m/3 + 1$ for security against active adversaries, thus this approach is very inefficient for a large number of tallying authorities.

The decryption circuit of Regev's scheme without message authentication is in $O(u\kappa)$. As demonstrated previously the introduced overhead is a factor in $O(m^2 t)$.

We conclude the examination by considering the preprocessing phase. Let $|C_{\text{MEnc}}|$ be the complexity of the encryption circuit of the underlying cryptosystem MPKE. For NTRU the cryptostystem MPKE has message space of size exponential in u and the protocol Π_{Smudge} for obtaining smudging values is in $O(m^2 tu\kappa^2 |C_{\text{MEnc}}|)$ for [BDOZ11], and in $O(mu\kappa |C_{\text{MEnc}}|)$ for [DPSZ12] respectively.

For Regev's Scheme the preprocessing necessary for each prime divisor of the modulus is in $O(\lceil u/\log\kappa\rceil m^2 tu\kappa |C_{\text{MEnc}}|)$, where the underlying cryptosystem MPKE has message space of size polynomial in κ, namely the size of the respective prime divisor.

Altogether we recommend NTRU with the SPDZ-protocol for actively secure decryption as the most efficient approach. This provides the additional advantage that the extension from an election scheme providing two options to an election scheme providing n options is straightforward, since the message space of NTRU consists of polynomials with n

coefficients. A major disadvantage is that the protocol is not guaranteed to terminate and cheating players cannot be detected.

After an abort or time-out we suggest to proceed depending on the required level of security. If the application allows to assume an honest majority, the decryption can be restarted with the protocol of [BDOZ11] for threshold decryption, where cheating players can be detected and excluded from the protocol. Alternatively the decryption can be restarted using general zero-knowledge techniques, where authorities have to proof that they are behaving according to the protocol. Note though that this introduces a major computational overhead.

Bibliography

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. "Fast cryptographic primitives and circular-secure encryption based on hard learning problems". In: *Advances in Cryptology-CRYPTO 2009*. Springer, 2009, pp. 595–618.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. "Multiparty computation with low communication, computation and interaction via threshold FHE". In: *Advances in Cryptology–EUROCRYPT 2012*. Springer, 2012, pp. 483–501.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. "Better zero-knowledge proofs for lattice encryption and their application to group signatures". In: *Advances in Cryptology–ASIACRYPT 2014*. Springer, 2014, pp. 551–572.

[BD10] Rikke Bendlin and Ivan Damgård. "Threshold Decryption and Zero-Knowledge Proofs for Lattice-Based Cryptosystems." In: *TCC*. Vol. 5978. Springer. 2010, pp. 201–218.

[BDO14] Carsten Baum, Ivan Damgård, and Claudio Orlandi. "Publicly auditable secure multi-party computation". In: *Security and Cryptography for Networks*. Springer, 2014, pp. 175–196.

[BDOZ10] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. *Semi-Homomorphic Encryption and Multiparty Computation*. Cryptology ePrint Archive, Report 2010/514. `http://eprint.iacr.org/`. 2010.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. "Semi-homomorphic encryption and multiparty computation". In: *Advances in Cryptology–EUROCRYPT 2011*. Springer, 2011, pp. 169–188.

[BT94] Josh Benaloh and Dwight Tuinstra. "Receipt-free secret-ballot elections". In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. ACM. 1994, pp. 544–553.

[Can01] Ran Canetti. "Universally composable security: A new paradigm for cryptographic protocols". In: *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE. 2001, pp. 136–145.

[Can04] Ran Canetti. *Universal Composability*. 6.897: Advanced Topics in Cryptography. `http://courses.csail.mit.edu/6.897/spring04/L1+2.pdf/`. 2004.

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. "Share conversion, pseudorandom secret-sharing and applications to secure computation". In: *Theory of Cryptography*. Springer, 2005, pp. 342–362.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper B Nielsen. *Multiparty computation from threshold homomorphic encryption*. Springer, 2001.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of partial knowledge and simplified design of witness hiding protocols". In: *Advances in Cryptology—CRYPTO'94*. Springer. 1994, pp. 174–187.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. "A secure and optimally efficient multi-authority election scheme". In: *European transactions on Telecommunications* 8.5 (1997), pp. 481–490.

[Dam15] Ivan Damgård. *The SPDZ Protocol, Part 1 and 2*. 5th Bar-Ilan Winter School on Cryptography. `https://www.youtube.com/watch?v=N80DV3Brds0/`. `https://www.youtube.com/watch?v=Ce45hp24b2E/`. 2015.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. "Practical covertly secure MPC for dishonest majority–or: Breaking the SPDZ limits". In: *Computer Security–ESORICS 2013*. Springer, 2013, pp. 1–18.

[DL12] Ivan Damgård and Adriana López-Alt. "Zero-knowledge proofs with low amortized communication from lattice assumptions". In: *Security and Cryptography for Networks*. Springer, 2012, pp. 38–56.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. "Multiparty computation from somewhat homomorphic encryption". In: *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 643–662.

[Fuh12] Paul A. Fuhrmann. *A Polynomial Approach to Linear Algebra*. Universitext. Springer New York, 2012. ISBN: 9781441987341. URL: `https://books.google.de/books?id=jenjBwAAQBAJ`.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. "NTRU: A ring-based public key cryptosystem". In: *Algorithmic number theory*. Springer, 1998, pp. 267–288.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In: *Journal of the ACM (JACM)* 60.6 (2013), p. 43.

[Lyu12] Vadim Lyubashevsky. "Lattice signatures without trapdoors". In: *Advances in Cryptology–EUROCRYPT 2012*. Springer, 2012, pp. 738–755.

[MM11] Daniele Micciancio and Petros Mol. "Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions". In: *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 465–484.

[MR08] Daniele Micciancio and Oded Regev. "Lattice-based Cryptography". In: (2008).

[Pei09] Chris Peikert. “Public-key cryptosystems from the worst-case shortest vector problem”. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM. 2009, pp. 333–342.

[Reg05] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. ACM. 2005, pp. 84–93.

[Reg09] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Journal of the ACM (JACM)* 56.6 (2009), p. 34.

[Sha79] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[Sho97] Peter W Shor. “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”. In: *SIAM journal on computing* 26.5 (1997), pp. 1484–1509.

[SS11] Damien Stehlé and Ron Steinfeld. “Making NTRU as secure as worst-case problems over ideal lattices”. In: *Advances in Cryptology–EUROCRYPT 2011*. Springer, 2011, pp. 27–47.