Institute of Information Security

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Verifiable Tally-Hiding E-Voting with Fully Homomorphic Encryption

Sebastian Hasler

Course of Study:	Informatik
Examiner:	Prof. Dr. Ralf Küsters
Supervisor:	Johannes Müller, M.Sc.
Commenced:	November 14, 2019
Completed:	May 14, 2020

Abstract

An E-voting system is *end-to-end verifiable* if arbitrary external parties can check whether the result of the election is correct or not. It is *tally-hiding* if it does not disclose the full election result but rather only the relevant information, such as e.g. the winner of the election.

In this thesis we pursue the goal of constructing an end-to-end verifiable tally-hiding E-voting system using fully homomorphic encryption. First we construct an alteration of the GSW levelled fully homomorphic encryption scheme based on the *learning with errors over rings* assumption. We utilize a key homomorphic property of this scheme in order to augment the scheme by a distributed key generation and distributed decryption. This leads to a passively secure 4-round multi-party computation protocol in the common random string model that can evaluate arithmetic circuits of arbitrary size. The complexity of this protocol is quasi-linear in the number of parties, polynomial in the security parameter and polynomial in the size of the circuit. By using Fiat-Shamir-transformed discrete-log-based zero-knowledge proofs we achieve security against active adversaries in the random oracle model while preserving the number of 4 rounds. Based on this actively secure protocol we construct an end-to-end verifiable tally-hiding E-voting system that has quasi-linear time complexity in the number of voters.

Contents

Acronyms

CRS common reference string. 11

FHE fully homomorphic encryption. 9

iff. if and only if. 12

IPS interactive proof system. 14

ITM interactive Turing machine. 11

LWE learning with errors. 9

MPC Multi-Party Computation. 9

PoK proof of knowledge. 14

ppt probabilistic and polynomial-time. 12

Ring-LWE learning with errors over rings. 9

ROM random oracle model. 10

SIVP shortest independent vectors problem. 21

VSD voter supporting device. 53

ZKPoK zero-knowledge proof of knowledge. 10

1 Introduction

In an E-voting system each voter creates an electronic ballot that contains the voter's choice in encrypted or encoded form and the ballots are tallied by computers. This has several advantages over tallying by humans. In classical voting where the tallying is done by hamans, those humans will always learn the full result of the election including the number of votes each candidate got. Learning the full result however might allow somebody to infer information about the individual choices of the voters, especially if the number of voters is small. If some candidate got no votes, then it is obvious that no voter chose that candidate. E-voting systems on the other hand can use sophisticated cryptography in order to ensure that only certain information such as the answer to "Who is the winner?" or "What is the order of the candidates?" is disclosed. This property is called *tally-hiding*. A further desired property is *end-to-end verifiability* which intuitively means that an arbitrary external party can check whether the result of the tallying is correct or not. E-voting has several other advantages for which we refer to [Mül19].

In modern E-voting systems the tallying might be done by shuffling the ballots using mix-nets and then decrypting them. This way it is kept secret which choice belongs to which voter. However, when using this approach it will typically still be public which candidate got how many votes, so everybody learns the full result of the election. Another possibility is to use homomorphic encryption. A homomorphic encryption scheme is an encryption scheme where we can perform computations on encrypted data. Suppose we have some function from the tuple of voters' choices to some *result space*. Such a function is called a *result function* and might for example output the winner of the election. If each voter's ballot is a homomorphic encryption of the voter's choice, then we can evaluate that function on the ballots (i.e. on the encrypted choices). If we decrypt the resulting ciphertext we obtain the same result as if we evaluated the function on the plaintext choices instead, but all intermediate values of the evaluation are kept secret.

There are different variants of homomorphic encryption such as *fully homomorphic encryption (FHE)* or *levelled FHE*. For the introduction of those terms we refer to [ABC+15].

In this work we construct a levelled FHE scheme based on the learning with errors over rings (Ring-LWE) assumption [LPR12]. Our levelled FHE scheme is the Ring-LWE analogue to the GSW FHE scheme [GSW13] which is based on the learning with errors (LWE) assumption [Reg05]. The GSW scheme and our scheme have the property that the sum of two key pairs is again a valid key pair. This key homomorphic property makes a distributed key generation possible, where each participating party obtains the public key and a share of the private key. Then everybody can encrypt ciphertexts but the parties can decrypt ciphertexts only collaboratively. The same thing was already done in [AJW11] for the BGV FHE scheme [BGV11]. Using this we construct a passively secure 4-round Multi-Party Computation (MPC) protocol in the common random string model [FF00].

We augment the protocol by short zero-knowledge proofs from [PLS19] in order to achieve security even against active adversaries. The zero-knowledge proofs are performed non-interactively using the Fiat-Shamir transform [FS86] so that all parties can verify the proof and the number of rounds

does not increase. This way we have an actively secure 4-round MPC protocol in the random oracle model (ROM) [BR93]. As those zero-knowledge proofs are discrete-log-based, it follows that our actively secure MPC protocol's security depends (additionally to the Ring-LWE assumption) on the discrete-log assumption, too. However, if discrete-log is broken in the future by e.g. practicable quantum computers, the privacy of past protocol executions is still preserved. That is, an adversary needs to take part in the protocol and have access to a quantum computer *during* the protocol execution in order to perform a successful discrete-log-based attack using a quantum computer.

Based on our MPC protocol we subsequently construct an end-to-end verifiable E-voting system (in the ROM) which can be instantiated with any result function that can be evaluated using the MPC protocol. In theory every result function is possible, because our MPC protocol works with arbitrary arithmetic circuits (of a certain type which does not constrain generality). However, in practice the parameters for the underlying levelled FHE scheme become quite large for deep circuits. For three families of tally-hiding result functions we construct the corresponding families of arithmetic circuits and analyze the asymptotic complexity of the resulting end-to-end verifiable tally-hiding E-voting system. Notably, for fixed security parameter we obtain time complexity in $\tilde{\Theta}(N)$ where N is the number of voters. This is optimal up to a polylogarithmic factor.

An experimental implementation of our actively secure MPC protocol which also supports the E-voting use case will be available at [Has20].

1.1 Roadmap

Chapter 2 briefly recalls the necessary cryptographic preliminaries, such as definitions of schemes and the corresponding security definitions. In Chapter 3 we construct our levelled FHE scheme and the passively secure version of our MPC protocol. We also analyze its asymptotic complexity depending on the security parameter, the number of parties and the depth of the evaluated arithmetic circuit. In Chapter 4 we describe the usage of non-interactive zero-knowledge proofs of knowledge in order to provide security against active adversaries, and we analyze how this changes the asymptotic complexity. Chapter 5 deals with verifiable tally-hiding E-voting based on our actively secure MPC protocol. Lastly, Chapter 6 concludes the thesis and provides an outlook on possible future work.

2 Preliminaries

In this chapter we briefly go over the necessary cryptographic preliminaries. Throughout this thesis we denote column vectors by bold lower case letters ($\mathbf{v}$) and matrices by bold upper case letters ($\mathbf{M}$). In this work the logarithm is always to the base 2. For any natural number n we denote by $[n]$ the set $[n] := \{1, \ldots, n\}$.

2.1 Computational Model

Multi-party protocols are formalized as tuples of interactive Turing machines (ITMs), often called parties. In this work, multiple connected ITMs are always connected via a secure broadcast channel and secure point-to-point channels. The broadcaster or sender of a message is always known to the recipient(s). Our protocols don't use the point-to-point channels, but adversaries might use them in order to collude. Therefore the protocols constructed in this work can be implemented even if only a broadcast channel is available. How to construct secure broadcast channels is not in the scope of this work. We always assume that the parties have access to a single secure broadcast channel, while in practice a decentralized solution to implement the broadcast channel might be used.

We will construct our passively secure MPC protocol in the common random string model [FF00]. The common random string model is a special case of the common reference string (CRS) model where the reference string consists of independent and uniformly distributed bits. The actively secure version will then be in the ROM [BR93].

The ROM is more powerful than the common random string model in the sense that every protocol that can be defined in the common random string model can also be defined in the ROM. This is because the ROM can be interpreted as that the parties have *random access* to an infinitely long tape of randomness. Therefore, when defining a protocol in the ROM, we can run sub-protocols that were initially defined in the common random string model. This common random string can be implemented as taking the randomness from the infinite sequence $H(0)H(1)H(2)\ldots$ or alternatively using a trusted setup.

2.2 Probability Spaces

When working with probability spaces, we denote by $\mathsf{Pr}[\cdot]$ the probability distribution and by $\exp\cdot$ the expected value.

Definition 2.2.1 (Probability space of a machine)**.** Let $\mathcal{A}$ be an arbitrary probabilistic Turing machine with runtime bound $t(\cdot)$, let x be an input and let $r \in \{0,1\}^{t(|x|)}$. Then we denote by $\mathcal{A}^r(x)$ the result of executing $\mathcal{A}$ on input x with random coins r. We call $\left(\Omega, 2^{\Omega}, \mathsf{Pr}\right)$ with $\Omega = \{0,1\}^{t(x)}$ and uniform probability distribution Pr the *probability space of* $\mathcal{A}(x)$. Over this probability space $\mathcal{A}(x)$ is a random variable defined by $\mathcal{A}(x)(r) := \mathcal{A}^r(x)$.

2.3 Indistinguishable Families of Random Variables

In order to define indistinguishability, we first provide some necessary definitions which can for example be found in textbooks [Vad12].

Definition 2.3.1 (negligible)**.** Let $f : \mathbb{N} \to \mathbb{R}$ be a function. f is *negligible* if and only if (iff.) for every polynomial p there exists $n_0 \in \mathbb{N}$ such that$\forall n \geq n_0 : f(n) \leq 1/p(n)$.

We often use a security parameter λ. If we say that some term $t(\lambda)$ (i.e. a term that depends on λ) is negligible, then we mean that the function $\lambda \mapsto t(\lambda)$ is negligible.

Definition 2.3.2 (Statistical difference [Vad12])**.** Let X and Y be two random variables with range U. The *statistical difference between X and Y* is defined as

$$\Delta(X,Y) := \frac{1}{2} \sum_{u \in U} |\mathsf{Pr}[X = u] - \mathsf{Pr}[Y = u]|.$$

Definition 2.3.3 (ε-close random variables [Vad12])**.** Let X and Y be two random variables with range U. We define that X and Y are *ε-close*, denoted by $X \approx_{\varepsilon} Y$, iff. $\Delta(X,Y) \leq \varepsilon$.

Definition 2.3.4 (Statistical indistinguishability)**.** Let $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ be two families of random variables on finite domains. X and Y are *statistically indistinguishable*, denoted by $X \equiv_s Y$, if there exists a negligible function $\varepsilon(\cdot)$ such that for for all $\lambda \in \mathbb{N}$ it holds true that $X_\lambda \approx_{\varepsilon(\lambda)} Y_\lambda$.

Definition 2.3.5 (Computational indistinguishability)**.** Let $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ be two families of random variables on finite domains. X and Y are *computationally indistinguishable*, denoted by $X \equiv_c Y$, if for all probabilistic and polynomial-time (ppt) Turing machines U with output range $\{0,1\}$ the advantage $\left|\mathsf{Exp}\left[U\left(1^\lambda, X_\lambda\right)\right] - \mathsf{Exp}\left[U\left(1^\lambda, Y_\lambda\right)\right]\right|$ is negligible.

The proof to the following lemma can be found in textbooks such as [KW11].

Lemma 2.3.1. *If $X \equiv_s Y$, then $X \equiv_c Y$.*

2.4 Arithmetic Circuits

In this work, an arithmetic circuit is a directed acyclic graph consisting of input, addition, multiplication, negation and output gates. Input gates have no predecessors. Addition and multiplication gates have two predecessors. Negation and output gates have one predecessor. And output gates have no successors while other gates can have arbitrarily many successors. The function computed by an arithmetic circuit is defined in the usual way. If the arithmetic circuit is obvious from the context, then we denote by N_{inputs} its number of input gates and by N_{outputs} its number of output gates.

Definition 2.4.1 (1-bounded arithmetic circuit)**.** An arithmetic circuit is 1-bounded if on input from $\{0,1\}^{N_{\mathsf{inputs}}}$ all intermediate values and output values are in $\{0,1\}$.

2.5 Cryptographic Primitives and their Security

In this section we give scheme definitions and security definitions for the cryptographic primitives used in this thesis.

Definition 2.5.1 (Commitment scheme)**.** A *commitment scheme* is a tuple $(\mathsf{Gen}, \mathsf{Com})$ of ppt algorithms of the following form.

- $\mathsf{Gen}\left(1^\lambda\right)$ takes the security parameter λ and outputs public parameters params, including the domain of $\mathsf{Com}(\mathsf{params}, \cdot)$.
- $\mathsf{Com}(\mathsf{params}, v)$ outputs a commitment for input v.

Alice can commit to some value v as follows. She computes $c = \mathsf{Com}^r(\mathsf{params}, v)$, where r are the random coins drawn during that computation. Then she sends c to Bob. Note that Com is a ppt algorithm and therefore committing to the same value v multiple times typically results in distinct commitments. When Alice likes to open the commitment, she sends (v, r) to Bob. Bob then verifies that c is indeed a commitment to v by checking whether $c = \mathsf{Com}^r(\mathsf{params}, v)$.

Definition 2.5.2 (Perfectly hiding)**.** A commitment scheme $(\mathsf{Gen}, \mathsf{Com})$ is *perfectly hiding* if for all $\lambda \in \mathbb{N}$, for all $\mathsf{params} \leftarrow \mathsf{Gen}\left(1^\lambda\right)$ and for all v_0, v_1 in the corresponding domain it holds true that

$$\mathsf{Com}(\mathsf{params}, v_0) \equiv \mathsf{Com}(\mathsf{params}, v_1).$$

Definition 2.5.3 (Computationally binding)**.** Let $\mathcal{K} = (\mathsf{Gen}, \mathsf{Com})$ be a commitment scheme. $\mathcal{K}$ is *computationally binding* if every ppt adversary $\mathcal{A}$ has negligible advantage $\mathsf{Adv}^{\text{binding}}_{\mathcal{A},\mathcal{K}}$ where $\mathsf{Adv}^{\text{binding}}_{\mathcal{A},\mathcal{K}}(\lambda) := \mathsf{Exp}\left[\mathbb{E}^{\text{binding}}_{\mathcal{A},\mathcal{K}}\left(1^\lambda\right)\right]$ is defined using the security game $\mathbb{E}^{\text{binding}}_{\mathcal{A},\mathcal{K}}$ presented in Algorithm 2.1

For our actively secure MPC protocol we will need the additional property that the used commitment scheme is *non-malleable*. For the definition of non-malleable we refer to [BGR+15]. Non-interactive non-malleable commitments are not possible in the standard model [GPR15] but they are possible in the ROM [BGR+15].

Algorithm 2.1 Security for computationally binding commitment schemes

procedure $\mathbb{E}^{\text{binding}}_{\mathcal{A},\mathcal{K}}(1^\lambda)$
  $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$
  $(v_0, r_0, v_1, r_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{params})$
  if $v_0 \neq v_1$ and $\mathsf{Com}^{r_0}(\mathsf{params}, v_0) = \mathsf{Com}^{r_1}(\mathsf{params}, v_1)$ **then**
    return 1
  else
    return 0
  end if
end procedure

Definition 2.5.4 (Interactive Turing machine)**.** An *ITM* is a probabilistic Turing machine that can send and receive messages. We denote the execution of two ITMs M_1 and M_2 on joint input x by $\langle M_1, M_2\rangle(x)$. During a run, the ITM that received the last message is always executing while the other ITM is blocked waiting for the next message. We define the probability space of $\langle M_1, M_2\rangle(x)$ analogously to Definition 2.2.1 using a pair of random coins.

Definition 2.5.5 (Interactive proof system)**.** Let P, V be a pair of ITMs where P has bounded runtime, V has polynomial runtime and V has output in $\{0,1\}$. Let $\langle P, V\rangle(x)$ denote the local output of V on joint input x. (P, V) is an *interactive proof system (IPS)* for a language L if there exist $c, s \in [0,1]$ such that:

- $s < c$.
- Completeness: $\forall x \in L : \mathsf{Pr}[\langle P, V\rangle(x) = 1] \geq c$.
- Soundness: For all bounded-runtime ITMs P^* it holds true that $\forall x \notin L : \mathsf{Pr}[\langle P, V\rangle(x) = 1] \leq s$.

Definition 2.5.6 (Perfect Zero-Knowledge)**.** Let (P, V) be an IPS for a language L. (P, V) is called *perfect zero-knowledge* if for all ppt ITMs V^* there exists a ppt Turing machine M^* such that

- $\mathsf{Pr}[M^*(x) = \bot] \leq \frac{1}{2}$,
- $\forall x, y : \mathsf{Pr}[\langle P, V\rangle(x) = y] = \mathsf{Pr}[M^*(x) = y \mid M^*(x) \neq \bot]$.

Definition 2.5.7 (Proof of Knowledge)**.** Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation that is decidable in polynomial time — i.e. $R \in \mathsf{P}$ — and let (P, V) be an IPS for $L_R := \{x \mid \exists y : |y| \leq f(|x|) \text{ and } (x, y) \in R\}$ where f is some polynomial. For any prover P^*, let $P^*_{r,x}(m)$ denote the message sent by P^{*r} — i.e. P^* using random coins r — on joint input x after past communication m. (That is, we can use $P^*_{r,x}$ as an oracle in order to have rewindable black box access to P^{*r}.) Let $\kappa : \mathbb{N} \to [0,1[$ be a function. (P, V) is called *proof of knowledge (PoK) with knowledge error* κ if:

- Non-triviality: $\forall x \in L_R : \mathsf{Pr}[\langle P, V\rangle(x) = 1] = 1$.
- Validity: There exists a polynomial q and an oracle ITM E such that for all $x \in L_R$, for all bounded-runtime ITMs P^* — let t be the runtime bound of P^* — and for all $r \in \{0,1\}^{t(|x|)}$ with $p(r, x) := \mathsf{Pr}[\langle P^{*r}, V\rangle(x) = 1] > \kappa(|x|)$ it holds true that:

– $\left(x, E^{P^*_{r,x}}(x)\right) \in R$.

– The expected runtime of $E^{P^*_{r,x}}(x)$ is less than $\frac{q(|x|)}{p(r,x)-\kappa(|x|)}$.

Note that the prover in an IPS has bounded runtime but does not necessarily run in polynomial time. For example, there is no known ppt prover for any NP-complete L_R as the contrary would imply BPP $\subseteq$ NP. Typically in a PoK used in cryptography the prover first computes the witness and then the remaining computation of the prover runs in polynomial time. Such a prover can be used as a building block in protocols that need to run in polynomial time. Thereby the prover is slightly modified such that it takes the witness as an auxiliary input and hence skips the computation of the witness.

By ZKPoK we designate an IPS that is perfect zero-knowledge and PoK.

Definition 2.5.8 (Public-key encryption scheme)**.** A *public-key encryption scheme* is a tuple $\mathcal{S} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where the following holds true.

- $\mathsf{Gen}(1^\lambda)$ is a ppt algorithm that outputs a key pair $(\mathsf{pk}, \mathsf{sk})$. We denote the range of Gen by $K_\mathcal{S}$ and define $K_\mathcal{S}^{\mathsf{pub}} := \{\mathsf{pk} \mid (\mathsf{pk}, \mathsf{sk}) \in K_\mathcal{S}\}$ and $K_\mathcal{S}^{\mathsf{sec}} := \{\mathsf{sk} \mid (\mathsf{pk}, \mathsf{sk}) \in K_\mathcal{S}\}$.
- $X = \left(X_{\mathsf{pk}}\right)_{\mathsf{pk} \in K_\mathcal{S}^{\mathsf{pub}}}$ is a family of (plaintext) sets.
- $\mathsf{Enc}\left(\mathsf{pk} : K_\mathcal{S}^{\mathsf{pub}}, x : \{0,1\}^*\right) : \{0,1\}^*$ is a ppt (encryption) algorithm.
- $\mathsf{Dec}\left(\mathsf{sk} : K_\mathcal{S}^{\mathsf{sec}}, y : \{0,1\}^*\right) : \{0,1\}^*$ is a deterministic polynomial-time (decryption) algorithm.
- For all $(\mathsf{pk}, \mathsf{sk}) \in K_\mathcal{S}$ and all $x \in X_{\mathsf{pk}}$ it holds true that $\mathsf{Pr}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, x)) = x] = 1$.

As a building block for proving the security of our MPC protocol, we will prove that the underlying encryption scheme satisfies the following security definition.

Definition 2.5.9 (Pseudo-random ciphertexts)**.** Let $\mathcal{S} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme and for any public key $\mathsf{pk} \in K_\mathcal{S}^{\mathsf{pub}}$ denote by Y_{pk} the corresponding ciphertext space, i.e. the range of $\mathsf{Enc}(\mathsf{pk}, \cdot)$. that has a fixed ciphertext space Y independent of the public key. $\mathcal{S}$ has *pseudo-random ciphertexts* if every ppt adversary $\mathcal{A} = (\mathcal{AF}, \mathcal{AG})$ where $\mathcal{AG}$ outputs $\beta' \in \{0,1\}$ has negligible advantage $\left|\mathsf{Adv}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}\right|$ where $\mathsf{Adv}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}(\lambda) := 1 - 2\mathsf{Exp}\left[\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}(1^\lambda)\right]$ is defined using the security game $\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}$ presented in Algorithm 2.2.

This notion of pseudo-random ciphertexts is also called *real-or-random* in the literature [KW11]. The reader might be more familiar with the so-called IND-CPA security which is defined as follows.

Definition 2.5.10 (IND-CPA)**.** A public-key encryption scheme $\mathcal{S} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ provides *indistinguishability under chosen plaintext attack (*IND-CPA *security)* if every ppt adversary $\mathcal{A} = (\mathcal{AF}, \mathcal{AG})$, where $\mathcal{AF}(1^\lambda, \mathsf{pk})$ outputs $((z_0, z_1), \alpha)$ with $z_0, z_1 \in X_{\mathsf{pk}}$ and $|z_0| = |z_1|$ and $\mathcal{AG}(1^\lambda, \alpha, y)$ outputs $\beta' \in \{0,1\}$, has negligible advantage $\left|\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathcal{S}}\right|$ where $\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathcal{S}}(\lambda) := 1 - 2\mathsf{Exp}\left[\mathbb{E}^{\text{IND-CPA}}_{\mathcal{A},\mathcal{S}}(1^\lambda)\right]$ is defined using the security game $\mathbb{E}^{\text{IND-CPA}}_{\mathcal{A},\mathcal{S}}$ presented in Algorithm 2.3.

Algorithm 2.2 Security game for pseudo-random ciphertexts

procedure $\mathbb{E}^{\text{PRC}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
  $(z, \alpha) \leftarrow \mathcal{AF}(1^\lambda, \mathsf{pk})$
  $\beta \xleftarrow{\$} \{0,1\}$
  if $\beta = 1$ **then**
    $y \leftarrow \mathsf{Enc}(\mathsf{pk}, z)$
  else
    $y \xleftarrow{\$} Y_{\mathsf{pk}}$
  end if
  $\beta' \leftarrow \mathcal{AG}(1^\lambda, \alpha, y)$
  return $1_{\beta=\beta'}$
end procedure

Algorithm 2.3 Security game for chosen plaintext attack

procedure $\mathbb{E}^{\text{IND-CPA}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
  $((z_0, z_1), \alpha) \leftarrow \mathcal{AF}(1^\lambda, \mathsf{pk})$
  $\beta \xleftarrow{\$} \{0,1\}$
  $y \leftarrow \mathsf{Enc}(\mathsf{pk}, z_\beta)$
  $\beta' \leftarrow \mathcal{AG}(1^\lambda, \alpha, y)$
  return $1_{\beta=\beta'}$
end procedure

The proof to the following lemma can be found in [KTV10] where pseudo-random ciphertexts is called real-or-random. In this work we will only use the fact that our encryption scheme has pseudo-random ciphertexts, so Lemma 2.5.1 is formally not needed.

Lemma 2.5.1. *If a public-key encryption scheme has pseudo-random ciphertexts, then it provides* IND-CPA *security.*

2.6 Multi-Party Computation

In this section we define MPC and also the notion of a threshold levelled FHE scheme, because we will construct our MPC protocol from such a scheme.

Definition 2.6.1 (Threshold C-evaluation scheme)**.** Let $N_{\mathsf{parties}} \in \mathbb{N}$ and let C be a family of 1-bounded arithmetic circuits. A *threshold C-evaluation scheme for N_{parties} parties* is a tuple $\mathcal{S} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ where the following holds true.

- Gen is a tuple of N_{parties} connected ppt ITMs. We denote by $\mathsf{Gen}(1^\lambda)$ the execution of those ITMs on joint input 1^λ. For $k \in [N_{\mathsf{parties}}]$ we associate the k^{th} ITM in that tuple with the party P_k. When running $\mathsf{Gen}(1^\lambda)$, each party P_k — $k \in [N_{\mathsf{parties}}]$ — has a local output $(\mathsf{pk}, \mathsf{sk}_k, \mathsf{evk})$. We require the pk and evk in the local outputs to match. This way we can use

the notation $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{Gen}\left(1^\lambda\right)$ where $\mathsf{sk} = (\mathsf{sk}_k)_{k\in[N_{\mathsf{parties}}]}$. Just as in Definition 2.5.8 we denote the range of $\mathsf{Gen}\left(1^\lambda\right)$ by $K_{\mathcal{S}}$ and define $K_{\mathcal{S}}^{\mathsf{pub}} := \{\mathsf{pk} \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \in K_{\mathcal{S}}\}$, $K_{\mathcal{S}}^{\mathsf{sec}} := \{\mathsf{sk} \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \in K_{\mathcal{S}}\}$. and $K_{\mathcal{S}}^{\mathsf{eval}} := \{\mathsf{evk} \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \in K_{\mathcal{S}}\}$.

- $X = \left(X_{\mathsf{pk}}\right)_{\mathsf{pk}\in K_{\mathcal{S}}^{\mathsf{pub}}}$ is a family of (plaintext) sets.
- $\mathsf{Enc}\left(\mathsf{pk} : K_{\mathcal{S}}^{\mathsf{pub}}, x : \{0,1\}^*\right) : \{0,1\}^*$ is a ppt (encryption) algorithm.
- $\mathsf{Eval}\left(\mathsf{evk} : K_{\mathcal{S}}^{\mathsf{eval}}, f : C, x : \{0,1\}^*\right) : \{0,1\}^*$ is a deterministic polynomial-time (evaluation) algorithm.
- Dec is a tuple of N_{parties} connected ppt ITMs. We denote by

  $$x' \leftarrow \mathsf{Dec}\left(\mathsf{sk} = (\mathsf{sk}_k)_{k\in[N_{\mathsf{parties}}]} : K_{,}^{\mathsf{sec}} y : \{0,1\}^*\right)$$

  the execution of those ITMs on joint input y where each party P_k — $k \in \left[N_{\mathsf{parties}}\right]$ — has the additional local input sk_k. When running $\mathsf{Dec}\left(\mathsf{sk} = (\mathsf{sk}_k)_{k\in[N_{\mathsf{parties}}]} : K_{,}^{\mathsf{sec}} y : \{0,1\}^*\right)$ each party has a local output x' and we require those local outputs to match.
- For all $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \in K_{\mathcal{S}}$, $f \in C$ and $x = \left(x_1, \ldots, x_{N_{\mathsf{inputs}}}\right) \in X_{\mathsf{pk}}^{N_{\mathsf{inputs}}}$ where N_{inputs} is the number of input gates of f, it holds true that

  $$\Pr\left[\mathsf{Dec}\left(\mathsf{sk}, \mathsf{Eval}\left(\mathsf{evk}, f, (\mathsf{Enc}(\mathsf{pk}, x_i))_{i\in[N_{\mathsf{inputs}}]}\right)\right) = f(x)\right] = 1.$$

It is possible to apply Definition 2.6.1 in various communication models. For our purposes, multiple connected ITMs are always connected via a secure broadcast channel and secure point-to-point channels, as described in Section 2.1.

Similarly to the definition of levelled FHE in the work of Armknecht et al. [ABC+15], we define the *threshold levelled FHE scheme* as a C-evaluation scheme with an additional depth parameter.

Definition 2.6.2 (Threshold levelled FHE scheme)**.** A *threshold levelled FHE scheme* is a tuple $\mathcal{S} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ where the following holds true.

- Gen differs from Definition 2.6.1 only in that it takes a depth parameter dpth as an additional joint input.
- For every $\mathsf{dpth} \in \mathbb{N}$ it holds true that $(X, \mathsf{Gen}(\cdot, \mathsf{dpth}), \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is a threshold C_{dpth}-evaluation scheme where C_{dpth} is the class of 1-bounded arithmetic circuits of depth at most dpth.

Definition 2.6.3 (MPC protocol)**.** Let $N_{\mathsf{parties}} \in \mathbb{N}$. An *MPC protocol* $\mathcal{P}$ for N_{parties} parties is a tuple $(P_1, \ldots, P_{N_{\mathsf{parties}}})$ of ITMs, called parties. $(y_k)_{k\in[N_{\mathsf{parties}}]} \leftarrow \mathcal{P}\left(1^\lambda, (x_k)_{k\in[N_{\mathsf{parties}}]}\right)$ denotes a protocol execution where each party P_k — $k \in [N_{\mathsf{parties}}]$ — takes input $\left(1^\lambda, x_k\right)$ and outputs y_k. If all parties produce equal output $y := y_1 = \cdots = y_{N_{\mathsf{parties}}}$, then we also write $y \leftarrow \mathcal{P}\left(1^\lambda, (x_k)_{k\in[N_{\mathsf{parties}}]}\right)$. We define the probability space of $\mathcal{P}\left(1^\lambda, (x_k)_{k\in[N_{\mathsf{parties}}]}\right)$ analogously to Definition 2.2.1 using an N_{parties}-tuple of random coins.

In the context of multi-party protocols, a *passive adversary* is an adversary that does not deviate from the protocol description. However, a passive adversary might choose its random coins in an adversarial fashion and multiple passive adversaries might collude in order to combine their information. An *active adversary* is simply an arbitrary ppt Turing machine.

Definition 2.6.4. Let $N \in \mathbb{N}$ and $I \subseteq [N]$. Define the projection $\pi_I(y_1, \ldots, y_N) := (y_i)_{i \in I}$. Hereby the $i \in I$ are sorted in ascending order.

Definition 2.6.5 (Correctness in the presence of passive adversaries)**.** Let $\mathcal{P}$ be an MPC protocol for N_{parties} parties and $f : \left(\{0,1\}^*\right)^{N_{\text{parties}}} \to \{0,1\}^*$. $\mathcal{P}$ provides *correctness in the presence of passive adversaries* with respect to f iff. for all $\lambda \in \mathbb{N}$ and for all input tuples x it holds true that $\mathsf{Pr}\left[\mathcal{P}\left(1^\lambda, x\right) \in \{f(x), \perp\}\right]$ is overwhelming in λ over the probability space of $\mathcal{P}\left(1^\lambda, x\right)$. That is, either all parties abort (denoted by the local output $\perp$) or all parties have the (correct) local output $f(x)$.

One might wonder why in the latter definition we don't account for passive adversaries. This is because any output of the honest parties in an execution of the protocol with passive adversaries might also occur in an honest execution. Therefore, if the above equality holds in all honest executions, then only the outputs of the adversaries can deviate in executions with passive adversaries. This is also why in the definition of correctness (Definition 2.6.7) we will only require correctness of the honest parties' local outputs.

Definition 2.6.6 (Computational privacy in the presence of passive adversaries)**.** Let $\mathcal{P} = \left(P_1, \ldots, P_{N_{\text{parties}}}\right)$ be an MPC protocol that provides correctness in the presence of passive adversaries with respect to a function f. $\mathcal{P}$ provides *computational privacy in the presence of passive adversaries* iff. for all $K \subseteq [N_{\text{parties}}]$ and for all tuples of ppt passive adversaries $\left(P_k^*\right)_{k \in K}$ there exists a ppt simulator $\mathcal{S}$ such that for all input tuples x

$$\left\{\mathcal{S}\left(1^\lambda, (x_k)_{k \in K}, f(x)\right)\right\}_{\lambda \in \mathbb{N}} \equiv_c \left\{\pi_K\left(\mathcal{P}^*\left(1^\lambda, (x_k)_{k \in [N_{\text{parties}}]}\right)\right)\right\}_{\lambda \in \mathbb{N}}.$$

Hereby, $\mathcal{P}^*$ is the MPC protocol derived from $\mathcal{P}$ by replacing P_k by P_k^* for each $k \in K$.

Note that for any MPC protocol $\mathcal{P}$ there can be at most one function f such that $\mathcal{P}$ provides correctness (in the presence of passive adversaries) with respect to f. Therefore perfect privacy (in the presence of passive adversaries) is well-defined, as it does not depend on the choice of f.

Definition 2.6.7 (Correctness)**.** Let $\mathcal{P}$ be an MPC protocol for N_{parties} parties and $f : \left(\{0,1\}^*\right)^{N_{\text{parties}}} \to \{0,1\}^*$. $\mathcal{P}$ provides *correctness* with respect to f iff. for all $\lambda \in \mathbb{N}$, for all input tuples x, for all $K \subseteq \left[N_{\text{parties}}\right]$ — let $H := \left[N_{\text{parties}}\right] \setminus K$ — and for all tuples of ppt adversaries $\left(P_k^*\right)_{k \in K}$ it holds true that $\mathsf{Pr}\left[\pi_H\left(\mathcal{P}^*\left(1^\lambda, x\right)\right) \in \left\{(f(x))^{|H|}, (\perp)^{|H|}\right\}\right]$ is overwhelming in λ over the probability space of $\mathcal{P}^*\left(1^\lambda, x\right)$. Hereby, $\mathcal{P}^*$ is the MPC protocol derived from $\mathcal{P}$ by replacing P_k by P_k^* for each $k \in K$. That is, either all honest parties abort or all honest parties have (correct) local output $f(x)$.

Definition 2.6.8 (Computational privacy)**.** An MPC protocol provides *computational privacy* if it satisfies Definition 2.6.6 but quantifying over all ppt adversaries instead of only the passive ones.

2.7 Number Theoretical Background

Before we state the Ring-LWE assumption, we need some preliminaries from algebraic number theory. For a Ring R and integer modulus $q \in \mathbb{Z}$, the quotient ring is denoted by $R_q := R/qR$ throughout this work.

Let d be a power of two. Then $\Phi_{2d}(X) = X^d + 1 \in \mathbb{Z}[X]$ is the so-called $2d^{\text{th}}$ cyclotomic polynomial. For a number a over some group, if $a^d = -1$, then a is a *primitive* $2d^{\text{th}}$ root of unity. Therefore the roots of $X^d + 1$ are primitive $2d^{\text{th}}$ roots of unity. Consider a prime modulus q with $q \equiv 1 \mod 2d$. Let $\mathbb{Z}_q^*$ be the multiplicative group of units of $\mathbb{Z}_q$. $\mathbb{Z}_q^*$ is a cyclic group of order $q-1$. Because $2d$ divides $q-1$, there is an element $\alpha \in \mathbb{Z}_q^*$ of order $2d$. Fix this element. For $i \in [d]$, let $\alpha_i = \alpha^{2i-1}$. We have $\gcd(2d, 2i-1) = 1$ and hence the α_i all have order $2d$. So, over $\mathbb{Z}_q$, we just found d roots of $X^d + 1$ and therefore $X^d + 1$ factors into linear polynomials modulo q. Specifically, $X^d + 1 = \prod_{i=1}^{d}(X - \alpha_i)$ in $\mathbb{Z}_q[X]$.

For $i \in [d]$ consider the ideals $\mathfrak{p}_i = \langle X - \alpha_i \rangle$ of $\mathbb{Z}_q[X]$. As the $X - \alpha_i$ are irreducible, they are pairwise coprime and Bézout's identity yields that, for $i_1, i_2 \in [d]$ and $i_1 \neq i_2$, there exists $g, h \in \mathbb{Z}_q[X]$ such that $g \cdot \left(X - \alpha_{i_1}\right) + h \cdot \left(X - \alpha_{i_2}\right) = 1 \in \mathfrak{p}_{i_1} + \mathfrak{p}_{i_2}$. It follows that the $\mathfrak{p}_i$ are pairwise coprime and therefore the Chinese remainder theorem applys, stating that we have the isomorphism

$$\begin{aligned} \mathbb{Z}_q[X]/\left\langle X^d + 1\right\rangle &\to \mathbb{Z}_q[X]/\mathfrak{p}_1 \times \cdots \times \mathbb{Z}_q[X]/\mathfrak{p}_d \\ f + \left\langle X^d + 1\right\rangle &\mapsto (f + \mathfrak{p}_1, \ldots, f + \mathfrak{p}_d). \end{aligned}$$

Given $f + \left\langle X^d + 1\right\rangle$, how can we compute a representation of $f + \mathfrak{p}_i$? It turns out that $X + \mathfrak{p}_i = X + \mathfrak{p}_i - (X - \alpha_i) = \alpha_i + \mathfrak{p}_i$ and inductively $X^k + \mathfrak{p}_i = \alpha_i^k + \mathfrak{p}_i$ for $k \in \mathbb{N}$. Hence every coset $f + \mathfrak{p}_i$ can be represented as $f + \mathfrak{p}_i = f(\alpha_i) + \mathfrak{p}_i$ where $f(\alpha_i) \in \mathbb{Z}_q$ is f evaluated at α_i. This also implies that $0 + \mathfrak{p}_i, \ldots, (q-1) + \mathfrak{p}_i$ is an exhausting list of elements of $\mathbb{Z}_q[X]/\mathfrak{p}_i$. Those q cosets of $\mathfrak{p}_i$ partition $\mathbb{Z}_q[X]/\left\langle X^d + 1\right\rangle$.

Definition 2.7.1 (Chinese remainder embedding)**.** We define the *Chinese remainder embedding* of $\mathbb{Z}_q[X]/\left\langle X^d + 1\right\rangle$ into $\mathbb{Z}_q$ by

$$\begin{aligned} \gamma : \mathbb{Z}_q[X]/\left\langle X^d + 1\right\rangle &\to \mathbb{Z}_q^d \\ f + \left\langle X^d + 1\right\rangle &\mapsto (f(\alpha_1), \ldots, f(\alpha_d)). \end{aligned}$$

The advantage of this embedding γ over the naive coefficient embedding is that in the range of γ both addition *and multiplication* are coordinate-wise [LPR13]. Algorithms are thus often more efficient when implemented using the Chinese remainder embedding instead of the coefficient embedding.

With this one might wonder why we even started with the interpretation as a polynomial ring. This is because the Ring-LWE problem which we formalize in Definition 2.8.1 needs an error distribution χ. It is only useful if the elements sampled from χ are small; that is, they have a small norm with respect to some embedding. Under the assumption that certain lattice problems are hard in the worst-case, Lyubashevsky et al. [LPR12] proved that Ring-LWE is hard for an error distribution that is a multivariate Gaussian distribution with respect to the canonical embedding, which we will define below. However, Ring-LWE is not hard when using the Chinese remainder embedding instead.

The Chinese remainder embedding is still useful as an optimization of computations over the ring.

Definition 2.7.2 (Canonical embedding)**.** For $i \in [d]$, let $\omega_i \in \mathbb{C}$ be the i^{th} complex root of $X^d + 1$ in some fixed order and let $\sigma_i : \mathbb{Z}[X]/\langle X^d + 1\rangle \to \mathbb{C}$ be the injective ring homomorphism defined by $X \mapsto \omega_i$. Then the *canonical embedding* of $\mathbb{Z}[X]/\langle X^d + 1\rangle$ to $\mathbb{C}^d$ is defined by

$$\begin{aligned}\sigma : \mathbb{Z}[X]/\langle X^d + 1\rangle &\to \mathbb{C}^d \\ f + \langle X^d + 1\rangle &\mapsto (f(\omega_1), \ldots, f(\omega_d)).\end{aligned}$$

As the σ_i are ring homomorphisms it follows that addition *and multiplication* are component-wise in the canonical embedding, too. Observe that a single component of $\sigma(f)$ is sufficient in order to uniquely define (and reconstruct) f, because the σ_i are injective. But on the other hand a single component of $\sigma(f)$ needs the same amount of storage space as a ring element in the coefficient embedding or Chinese remainder embedding. We preserve the convention of Lyubashevsky et al. [LPR12] and use the full canonical embedding (instead of just a single component) in our analysis. For implementing computations over the ring, the Chinese remainder embedding is more efficient.

Definition 2.7.3. Let $q \in \mathbb{N}$. We define the infinity norm

$$\begin{aligned}||\cdot||_\infty : \quad \mathbb{Z}_q^{m\times n} &\to \{0, \ldots, \lfloor q/2 \rfloor\} \\ (a_{i,j})_{i\in[m],j\in[n]} &\mapsto \max_{i\in[m],j\in[n]} |a_{i,j}|\end{aligned}$$

whereby each entry $a_{i,j} \in \{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$. Let $\Phi(X) \in \mathbb{Z}[X]$ be a polynomial and $R := \mathbb{Z}[X]/\langle \Phi(X)\rangle$. We extend $||\cdot||_\infty$ to ring elements and matrices over R_q as follows.

$$\begin{aligned}||\cdot||_\infty : \quad R_q^{m\times n} &\to \{0, \ldots, \lfloor q/2 \rfloor\} \\ (a_{i,j})_{i\in[m],j\in[n]} &\mapsto \max_{i\in[m],j\in[n]} \left|\left|\sigma(a_{i,j})\right|\right|_\infty\end{aligned}$$

Note that the infinity norm of ring elements is in Definition 2.7.3 defined with respect to the canonical embedding. We refer to [LPR13] for the properties of this infinity norm.

Definition 2.7.4. Let $\Phi(X) \in \mathbb{Z}[X]$ be a polynomial of degree d and $R := \mathbb{Z}[X]/\langle \Phi(X)\rangle$. For elements $r = c_0 X^0 + \cdots + c_{d-1} X^{d-1} \in R$ we define

$$\mathsf{coefficents}(r) := (c_0, \ldots, c_{d-1}).$$

For any interval I, we denote by R_I the set $R_I := \left\{x \in R \;\middle|\; \mathsf{coefficents}(x) \in I^d\right\}$; that is, the set of elements of R_q that have coordinates in I with respect to the coefficient embedding.

2.8 The Learning with Errors over Rings Assumption

The security of our levelled FHE scheme is based on the Ring-LWE assumption which was introduced in [LPR12].

Definition 2.8.1 (Ring-LWE assumption)**.** For security parameter λ, let $d(\lambda) \in \mathbb{N}$ and let $R = \mathbb{Z}[X]/\langle X^d + 1\rangle$. Let $q(\lambda)$ be an integer modulus. Let $\chi(\lambda)$ be a probability distribution over R_q. Let $\mathsf{Adv}^{\mathsf{RLWE}}_{\mathcal{A},d,q,\chi,m}(\lambda) := 1 - 2\mathsf{Exp}\left[\mathbb{E}^{\mathsf{RLWE}}_{\mathcal{A},d,q,\chi,m}(1^\lambda)\right]$ be defined using the security game $\mathbb{E}^{\mathsf{RLWE}}_{\mathcal{A},d,q,\chi,m}$ presented in Algorithm 2.4. The $\mathsf{RLWE}_{d,q,\chi}$ assumption states that for any ppt adversary $\mathcal{A}$ and $m \in \mathsf{poly}(\lambda)$ the advantage $\left|\mathsf{Adv}^{\mathsf{RLWE}}_{\mathcal{A},d,q,\chi,m}\right|$ is negligible.

Algorithm 2.4 Security game for Ring-LWE

procedure $\mathbb{E}^{\mathsf{RLWE}}_{\mathcal{A},d,q,\chi,m}(1^\lambda)$
  $\mathbf{a} \xleftarrow{\$} R_q^m$
  $\beta \xleftarrow{\$} \{0,1\}$
  if $\beta = 1$ **then**
    $r \xleftarrow{\$} \chi$; $\mathbf{e} \leftarrow \chi^m$; $\mathbf{b} := \mathbf{a} \cdot r + \mathbf{e}$
  else
    $\mathbf{b} \xleftarrow{\$} R_q^m$
  end if
  $\beta' \leftarrow \mathcal{A}(1^\lambda, \mathbf{a}, \mathbf{b})$
  return $1_{\beta=\beta'}$
end procedure

In the original definition in [LPR12] r was drawn uniformly at random from R_q. But Ring-LWE is equally hard when drawing r from the error distribution instead [LPR12], analogously to standard LWE [ACPS09].

Ring-LWE is useful because, similarly to standard LWE, the security of Ring-LWE-based cryptography can be based on the worst-case hardness of certain lattice problems. When Lyubashevsky et al. introduced Ring-LWE [LPR12], they provided a reduction from the shortest independent vectors problem (SIVP) over rings to Ring-LWE. They first reduced the SIVP to the the search version of Ring-LWE via a quantum reduction and then reduced the search version to the decision version. (A ppt algorithm $\mathcal{A}$ solves the decision version of Ring-LWE if it has non-negligible advantage $\mathsf{Adv}^{\mathsf{RLWE}}_{\mathcal{A},d,q,\chi,m}(\lambda)$.) By transitivity those two reductions lead to the following theorem.

Definition 2.8.2 (B-bounded distribution)**.** Let $B, d \in \mathbb{N}$ and let $R := \mathbb{Z}[X]/\langle X^d + 1\rangle$. A distribution χ over R_q is called B-bounded if for all values $r \xleftarrow{\chi} R_q$ sampled according to χ it holds true that $||r||_\infty < B$. We also write $r \leftarrow \chi$ instead of $r \xleftarrow{\chi} R_q$.

Theorem 2.8.1. *Let $d(\lambda)$ be a power of two, let $R := \mathbb{Z}[X]/\langle X^d + 1\rangle$ be the $2d^{th}$ cyclotomic polynomial, let $q(\lambda) \equiv 1 \mod d$ be a prime modulus and let $B(\lambda) \in \omega\left(\sqrt{\log d}\right)$ with $B \leq \sqrt{d \log d}$. Then there is an efficiently sampleable B-bounded distribution $\chi(\lambda)$ such that if the $\mathsf{RLWE}_{d,q,\chi}$ assumption does not hold true, then there exists a ppt quantum algorithm for the $\tilde{O}\left(q/B\sqrt{d}\right)$-approximate SIVP on ideal lattices over R.*

Note that this reduction can take arbitrary ideal lattices over R and therefore the security of Ring-LWE based cryptography is based on the worst-case hardness. The SIVP on ideal lattices over R is assumed to be hard if $d \in \Omega\big(\lambda \log(q/B)\big)$ [BGV11].

3 Passively Secure Multi-Party Computation

In this chapter we construct our levelled FHE scheme and the passively secure version of our MPC protocol. Afterwards in Section 3.6 we analyze its asymptotic complexity depending on the security parameter, the number of parties and the depth of the evaluated arithmetic circuit.

The GSW scheme by Gentry et al. [GSW13] makes use of bit decomposition of vectors and of a gadget matrix that is the inverse of bit decomposition. For our ring variant of the GSW scheme we use the analogues over the polynomial ring. We adopt notation by Alperin-Sherrif and Peikert [AP14].

Definition 3.0.1. Let $d \in \mathbb{N}$ be a power of two and let $R = \mathbb{Z}[X]/\langle X^d + 1\rangle$. Let $q, m \in \mathbb{N}$. Let $\mathbf{g} := \left(2^0, \ldots, 2^{\lfloor \log q \rfloor}\right)^T$. We define the gadget matrix

$$\mathbf{G} := \begin{bmatrix} \mathbf{g} & \\ & \mathbf{g} \end{bmatrix} \in R_q^{2(\lfloor \log q \rfloor + 1) \times 2}.$$

The empty cells are filled with zeros. Furthermore, for any ring element $r \in R_q$, let $\mathsf{BitDecomp}(r) \in R_{[0,1]}^{\lfloor \log q \rfloor + 1}$ denote the bit decomposition of r. That is, we have $\langle \mathsf{BitDecomp}(r), \mathbf{g} \rangle = r$. We define the function

$$\begin{aligned} \mathbf{G}^{-1} : \quad & R_q^{m \times 2} \to R_{[0,1]}^{m \times 2(\lfloor \log q \rfloor + 1)} \\ & \begin{pmatrix} a_{1,1} & a_{1,2} \\ \vdots & \vdots \\ a_{m,1} & a_{m,2} \end{pmatrix} \mapsto \begin{bmatrix} \mathsf{BitDecomp}(a_{1,1})^T & \mathsf{BitDecomp}(a_{1,2})^T \\ \vdots & \vdots \\ \mathsf{BitDecomp}(a_{m,1})^T & \mathsf{BitDecomp}(a_{m,2})^T \end{bmatrix}. \end{aligned} \tag{3.1}$$

Note that $\mathbf{G}^{-1}$ (which performs bit decomposition on matrices) itself is not a matrix but an efficiently computable function. The parameters d, q, m are always obvious from the context when we use $\mathbf{G}$ or $\mathbf{G}^{-1}$,

Lemma 3.0.1. *Let R, q and m be as in Definition 3.0.1 and let $\mathbf{M} \in R_q^{m \times 2}$. Then $\mathbf{G}^{-1}(\mathbf{M})\,\mathbf{G} = \mathbf{M}$.*

Proof. Let

$$\mathbf{M} =: \begin{pmatrix} a_{1,1} & a_{1,2} \\ \vdots & \vdots \\ a_{m,1} & a_{m,2} \end{pmatrix}.$$

For any $i \in \{1, \ldots, m\}$ the i^{th} row of $\mathbf{G}^{-1}(\mathbf{M})\,\mathbf{G}$ equals

$$\begin{aligned} &\begin{bmatrix} \mathsf{BitDecomp}(a_{i,1})^T & \mathsf{BitDecomp}(a_{i,2})^T \end{bmatrix} \mathbf{G} \\ &\qquad = \left(\langle \mathsf{BitDecomp}(a_{i,1}), \mathbf{g} \rangle, \langle \mathsf{BitDecomp}(a_{i,2}), \mathbf{g} \rangle\right) = (a_{i,1}, a_{i,2}) \end{aligned}$$

which is the i^{th} row of $\mathbf{M}$. □

3.1 The Ring-GSW Homomorphic Encryption Scheme

In this section we lay the foundation by constructing our levelled FHE scheme and proving that it has pseudo-random ciphertexts.

The GSW scheme can be transformed into a Ring-LWE variant with only one conceptual difference on which we elaborate further below. In [GSW13] the authors did not formulate the Ring-LWE variant, because at the time there were already better known Ring-LWE based levelled FHE schemes such as the BGV scheme [BGV11]. However, we do not want to use the BGV scheme because it is much more complicated due to the modulus reduction and nesting of keys. The GSW scheme is much easier to implement.

We now describe the resulting scheme. In our description the usage of $\mathbf{G}$ and $\mathbf{G}^{-1}$ is as proposed by Alperin-Sherrif and Peikert and differs from the original description by Gentry et al. [GSW13]. As a result, the ciphertexts also have a different form than in the original description. However, this is an entirely syntactic difference [AP14]. Ciphertexts of both variants can be converted losslessly into each other without knowing the secret key.

Definition 3.1.1 (Ring-GSW public-key encryption scheme)**.** For security parameter λ, let d, R, q, χ be as in Definition 2.8.1. Set $m = m(\lambda) := 2(\lfloor \log q \rfloor + 1)$. The Ring-$\mathsf{GSW}_{d,q,\chi}$ public-key encryption scheme is the tuple $\mathcal{S} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows. Thereby all algorithms are implicity parameterized on d, R, q, χ.

- $X = \left(X_{\mathsf{pk}}\right)_{\mathsf{pk} \in K_{\mathcal{S}}^{\mathsf{pub}}}$: For every public key $\mathsf{pk} \in K_{\mathcal{S}}^{\mathsf{pub}}$ the plaintext space is $X_{\mathsf{pk}} = \{0,1\}$.
- $\mathsf{Gen}\left(1^\lambda\right)$: Sample $r \xleftarrow{\$} \chi$ and set the secret key $\mathbf{s} := (1, -r)^T$. Sample $\mathbf{a} \xleftarrow{\$} R_q^m$; $\mathbf{e} \leftarrow \chi^m$, set $\mathbf{b} := \mathbf{a} \cdot r + \mathbf{e}$ and set the public key $\mathbf{A} := \begin{bmatrix}\mathbf{b} & \mathbf{a}\end{bmatrix}$. Return $(\mathsf{pk} = \mathbf{A}, \mathsf{sk} = \mathbf{s})$.
- $\mathsf{Enc}\Big(\mathsf{pk} = \mathbf{A} : R_q^{m\times 2},\ \mu : \{0,1\}\Big)$: Sample $t \leftarrow \chi$; $\mathbf{F} \leftarrow \chi^{m\times 2}$ and return $\mathbf{C} := t\mathbf{A} + \mathbf{F} + \mu\mathbf{G}$.
- $\mathsf{Dec}\Big(\mathsf{sk} = \mathbf{s} : R_q^2,\ \mathbf{C} : R_q^{m\times 2}\Big)$: With $i := \lfloor \log q \rfloor$, let $\mathbf{C}[i]$ be the i^{th} row (starting at 1) of $\mathbf{C}$ and compute the *pre-plaintext* $\mathsf{pp} := \langle \mathbf{C}[i], \mathbf{s} \rangle$. Return $\mu' := \left\lfloor ||\mathsf{pp}||_\infty / 2^{i-1} \right\rceil$.

While the original GSW scheme used Regev encryptions [Reg05] of zero in order to hide the term $\mu\mathbf{G}$ in a ciphertext, we can not use them, because the application of the leftover hash lemma [HILL99; ILL89] in Regev's security proof does not generalize to Ring-LWE. Therefore we use LPR encryptions [LPR12] of zero instead: Observe that each row of $t\mathbf{A} + \mathbf{F}$ is an LPR encryption of zero. This is the only conceptual difference between the original GSW scheme and Ring-GSW. We did not come up with this idea, but rather it was already mentioned in the appendix of [GSW13].

First we give an intuition, why decryption is correct, i.e. for all $\mu \in \{0,1\}$ it always holds true that $\mathsf{Dec}(\mathbf{s}, \mathsf{Enc}(\mathbf{A}, \mu)) = \mu$ where $(\mathbf{s}, \mathbf{A}) \leftarrow \mathsf{Gen}\left(1^\lambda\right)$. Observe that $\mathbf{As} = \mathbf{b} - \mathbf{a} \cdot r = \mathbf{e} \approx \mathbf{0}$ and that $\mathbf{Fs} \approx \mathbf{0}$. Hence for a ciphertext $\mathbf{C}$ we obtain

$$\mathbf{Cs} = t\mathbf{As} + \mathbf{Fs} + \mu\mathbf{Gs} = t\mathbf{e} + \mathbf{Fs} + \mu\mathbf{Gs} \approx \mu \begin{bmatrix} \mathbf{g} \\ -\mathbf{g} \cdot r \end{bmatrix}$$

with $\mathbf{g} = (2^0, 2^1, \ldots, 2^i)^T$. Taking the i^{th} row of that (approximate) equation yields $\mathsf{pp} \approx \mu 2^{i-1}$. If this held true with equality, then would $\mathsf{pp} \in \{0, 2^{i-1}\}$ and thus $\mu = \mu'$. Consequently decryption is correct if the error in this approximation is not too large. We also call this error *noise* and quantify it via the following definition.

Definition 3.1.2 (Noise vector)**.** For a Ring-GSW-ciphertext $\mathbf{C}$ that encrypts μ under secret key $\mathbf{s}$, we define $\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C}) := (\mathbf{C} - \mu\mathbf{G})\,\mathbf{s} \in R_q^m$ to be the *noise vector* of $\mathbf{C}$. We also write $\mathsf{noise}(\mathbf{C})$ if $\mathbf{s}$ and μ are obvious from the context.

We refer to the infinity norm of the noise as the *noise level* of the ciphertext.

Lemma 3.1.1. *If* $\left\|\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C})\right\|_\infty < 2^{\lfloor \log q \rfloor - 2}$*, then* Ring-$\mathsf{GSW}_{d,q,\chi}$ *decryption is correct, i.e.* $\mathsf{Dec}(\mathbf{s}, C) = \mu$.

Proof. Let $i = \lfloor \log q \rfloor$. Observe $\left\|\mathsf{pp} - \mu 2^{i-1}\right\|_\infty \leq ||\mathbf{Cs} - \mu\mathbf{Gs}||_\infty = ||\mathsf{noise}(\mathbf{C})||_\infty < 2^{i-2}$. For $\mu = 0$ it follows that $||\mathsf{pp}||_\infty / 2^{i-1} < \frac{1}{2}$. For $\mu = 1$, using the rearranged triangle inequality $||\mathsf{pp}||_\infty \geq \left\|2^{i-1}\right\|_\infty - \left\|2^{i-1} - \mathsf{pp}\right\|_\infty$, it follows that $||\mathsf{pp}||_\infty / 2^{i-1} \geq \left(\left\|2^{i-1}\right\|_\infty - \left\|\mathsf{pp} - 2^{i-1}\right\|_\infty\right) / 2^{i-1} > 1 - \frac{1}{2} = \frac{1}{2}$. Hence $\mu' = \left\lfloor ||\mathsf{pp}||_\infty / 2^{i-1} \right\rceil = \mu$. □

For a fresh ciphertext $\mathbf{C} = t\mathbf{A} + \mathbf{F} + \mu\mathbf{G}$, the noise is $(t\mathbf{A} + \mathbf{F})\,\mathbf{s} = t\mathbf{e} + \mathbf{Fs}$, where $\mathbf{e}$ is the error vector used during key generation and $t, \mathbf{F}$ are the errors used during encryption. These errors as well as the secret key $\mathbf{s}$ are sampled from the error distribution χ, so decryption is correct if the polynomials sampled from χ are not too large.

In terms of security we will prove that under certain assumptions Ring-GSW has pseudo-random ciphertexts.

Theorem 3.1.1. *If the* $\mathsf{RLWE}_{d,q,\chi}$ *assumption holds true, then* Ring-$\mathsf{GSW}_{d,q,\chi}$ *has pseudo-random ciphertexts (as defined in Definition 2.5.9).*

Proof. In this proof we use the conditional expectation which is for a random variable V and an event E defined by $\mathsf{Exp}[V \mid E] := \sum_{\omega \in E} V(\omega)/|E|$. Let $\mathcal{S} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) :=$ Ring-$\mathsf{GSW}_{d,q,\chi}$ and let $\mathcal{A} = (\mathcal{AF}, \mathcal{AG})$ be an arbitrary ppt adversary under the constraints of Definition 2.5.9. Observe that over the probability space of $\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$ it holds true that

$$
\begin{aligned}
\mathsf{Adv}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}(\lambda) &= 1 - 2\mathsf{Exp}\left[\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}\left(1^\lambda\right)\right] \\
&= 1 - 2\mathsf{Pr}\left[\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}\left(1^\lambda\right) = 1\right] \\
&= 1 - 2(\mathsf{Pr}[\beta' = \beta = 0] + \mathsf{Pr}[\beta' = \beta = 1]) \\
&= 1 - 2(\mathsf{Pr}[\beta' = 0 \mid \beta = 0] \cdot \underbrace{\mathsf{Pr}[\beta = 0]}_{=1/2} + \mathsf{Pr}[\beta' = 1 \mid \beta = 1] \cdot \underbrace{\mathsf{Pr}[\beta = 1]}_{=1/2}) \\
&= 1 - (\mathsf{Pr}[\beta' = 0 \mid \beta = 0] + \mathsf{Pr}[\beta' = 1 \mid \beta = 1]) \\
&= (1 - \mathsf{Pr}[\beta' = 0 \mid \beta = 0]) - \mathsf{Pr}[\beta' = 1 \mid \beta = 1] \\
&= \mathsf{Pr}[\beta' = 1 \mid \beta = 0] - \mathsf{Pr}[\beta' = 1 \mid \beta = 1] \\
&= \mathsf{Exp}[\beta' \mid \beta = 0] - \mathsf{Exp}[\beta' \mid \beta = 1].
\end{aligned} \tag{3.2}
$$

Thus it suffices to prove that $|\mathsf{Exp}[\beta' \mid \beta = 0] - \mathsf{Exp}[\beta' \mid \beta = 1]|$ is negligible in λ. To do so we give a sequence of games where the first has expected value $\mathsf{Exp}[\beta' \mid \beta = 0]$, the last has expected value $\mathsf{Exp}[\beta' \mid \beta = 1]$ and the difference between the expected values of consecutive games is negligible.

$\textsc{Game}_1$ is presented in Algorithm 3.1. On input 1^λ it simulates $\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}\langle\beta = 0\rangle(1^\lambda)$ and outputs β'. Therefore

$$\mathsf{Exp}\left[\textsc{Game}_1\left(1^\lambda\right)\right] = \mathsf{Exp}[\beta' \mid \beta = 0]$$

where β, β' are the corresponding random variables over the probability space of $\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$.

Algorithm 3.1 $\textsc{Game}_1$ for Ring-GSW

procedure $\textsc{Game}_1(1^\lambda)$
  $(\mathbf{A}, \mathbf{s}) \leftarrow \mathsf{Gen}(1^\lambda)$
  $(\mu, \alpha) \leftarrow \mathcal{AF}(1^\lambda, \mathbf{A})$
  $\beta = 0$
  if $\beta = 1$ **then**
    $\mathbf{C} \leftarrow \mathsf{Enc}(\mathbf{A}, \mu)$
  else
    $\mathbf{C} \xleftarrow{\$} R_q^{m\times 2}$
  end if
  $\beta' \leftarrow \mathcal{AG}(1^\lambda, \alpha, \mathbf{C})$
  return β'
end procedure

$\textsc{Game}_2$ (see Algorithm 3.2) is a purely syntactic modification of $\textsc{Game}_1$. The subprocedure $\mathsf{Gen}(1^\lambda)$ was inlined and the if-condition was optimized away. $\mathbf{s}$ was omitted, too, because it's not used after assignment. Obviously $\mathsf{Exp}\left[\textsc{Game}_1(1^\lambda)\right] = \mathsf{Exp}\left[\textsc{Game}_2(1^\lambda)\right]$.

Algorithm 3.2 $\textsc{Game}_2$ for Ring-GSW

procedure $\textsc{Game}_2(1^\lambda)$
  $\mathbf{a} \xleftarrow{\$} R_q^m$
  $r \leftarrow \chi$
  $\mathbf{e} \leftarrow \chi^m$
  $\mathbf{b} := \mathbf{a} \cdot r + \mathbf{e}$
  $\mathbf{A} := \begin{bmatrix}\mathbf{b} & \mathbf{a}\end{bmatrix}$
  $(\mu, \alpha) \leftarrow \mathcal{AF}(1^\lambda, \mathbf{A})$
  $\mathbf{C} \xleftarrow{\$} R_q^{m\times 2}$
  $\beta' \leftarrow \mathcal{AG}(1^\lambda, \alpha, \mathbf{C})$
  return β'
end procedure

Let $\mathcal{A}'$ be the ppt algorithm that, on input $(1^\lambda, \tilde{\mathbf{a}}, \tilde{\mathbf{b}})$, computes $\textsc{Game}_2\langle\mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b} = \tilde{\mathbf{b}}\rangle(1^\lambda)$. Observe that $\textsc{Game}_2$ on input 1^λ simulates the Ring-LWE security game $\mathbb{E}^{\mathsf{RLWE}}_{\mathcal{A}',d,q,\chi,m}\langle\beta = 1\rangle(1^\lambda)$, as defined in Algorithm 2.4, and outputs the β' of that game. That is,

$$\mathsf{Exp}\left[\textsc{Game}_2\left(1^\lambda\right)\right] = \mathsf{Exp}[\beta' \mid \beta = 1]$$

where β, β' are now the corresponding random variables over the probability space of $\mathbb{E}^{\mathsf{RLWE}}_{\mathcal{A}',d,q,\chi,m}(1^\lambda)$. By a calculation analogous to Equation (3.2), the $\mathsf{RLWE}_{d,q,\chi}$ assumption states that $|\mathsf{Exp}[\beta' \mid \beta = 0] - \mathsf{Exp}[\beta' \mid \beta = 1]|$ is negligible in λ. Analogously, $\textsc{Game}_3$ is a game with

$$\mathsf{Exp}\left[\textsc{Game}_3\left(1^\lambda\right)\right] = \mathsf{Exp}[\beta' \mid \beta = 0]$$

and hence we infer that $\left|\mathsf{Exp}\left[\textsc{Game}_3\left(1^\lambda\right)\right] - \mathsf{Exp}\left[\textsc{Game}_2\left(1^\lambda\right)\right]\right|$ is negligible.

Algorithm 3.3 $\textsc{Game}_3$ for Ring-GSW

procedure $\textsc{Game}_3(1^\lambda)$
  $\mathbf{a} \xleftarrow{\$} R_q^m$
  $\mathbf{b} \xleftarrow{\$} R_q^m$
  $\mathbf{A} := \begin{bmatrix}\mathbf{b} & \mathbf{a}\end{bmatrix}$
  $(\mu, \alpha) \leftarrow \mathcal{AF}\left(1^\lambda, \mathbf{A}\right)$
  $\mathbf{C} \xleftarrow{\$} R_q^{m\times 2}$
  $\beta' \leftarrow \mathcal{AG}\left(1^\lambda, \alpha, \mathbf{C}\right)$
  return β'
end procedure

In $\textsc{Game}_4$ presented in Algorithm 3.4, instead of drawing the two columns of $\mathbf{A}$ separately, we draw $\mathbf{A}$ uniformly at random from $R_q^{m\times 2}$ in one syntactic statement. Furthermore, we add $\mu\mathbf{G}$ to $\mathbf{C}$, but as $\mathbf{C}'$ is distributed uniformly at random, $\mathbf{C}' + \mu\mathbf{G}$ is still distributed uniformly at random. Therefore, both of these are purely syntactic changes of $\textsc{Game}_3$ and thus $\mathsf{Exp}\left[\textsc{Game}_3\left(1^\lambda\right)\right] = \mathsf{Exp}\left[\textsc{Game}_4\left(1^\lambda\right)\right]$.

Algorithm 3.4 $\textsc{Game}_4$ for Ring-GSW

procedure $\textsc{Game}_4(1^\lambda)$
  $\mathbf{A} \xleftarrow{\$} R_q^{m\times 2}$
  $\mathbf{C}' \xleftarrow{\$} R_q^{m\times 2}$
  $(\mu, \alpha) \leftarrow \mathcal{AF}\left(1^\lambda, \mathbf{A}\right)$
  $\mathbf{C} := \mathbf{C}' + \mu\mathbf{G}$
  $\beta' \leftarrow \mathcal{AG}\left(1^\lambda, \alpha, \mathbf{C}\right)$
  return β'
end procedure

In $\textsc{Game}_4$, interpret $\mathbf{A}$ and $\mathbf{C}'$ as column vectors in R_q^{2m}. Analogously to the next-to-last step, we can now replace $\mathbf{C}'$ by a vector $\mathbf{A} \cdot t + \mathbf{F}$ of Ring-LWE samples. This way we obtain $\textsc{Game}_5$ shown in Algorithm 3.5, with negligible $\left|\mathsf{Exp}\left[\textsc{Game}_5\left(1^\lambda\right)\right] - \mathsf{Exp}\left[\textsc{Game}_4\left(1^\lambda\right)\right]\right|$.

$\textsc{Game}_6$ is presented in Algorithm 3.6. Recall that $\mathsf{Enc}(\mathbf{A}, \mu) = t\mathbf{A} + \mathbf{F} + \mu\mathbf{G}$, so the step from $\textsc{Game}_5$ to $\textsc{Game}_6$ is again a purely syntactic change; i.e. $\mathsf{Exp}\left[\textsc{Game}_5\left(1^\lambda\right)\right] = \mathsf{Exp}\left[\textsc{Game}_6\left(1^\lambda\right)\right]$.

From $\textsc{Game}_6$ to $\textsc{Game}_7$ (see Algorithm 3.7) we performed the modifications from $\textsc{Game}_1$ to $\textsc{Game}_3$ in reverse. Analogous to those steps we can argue that $\left|\mathsf{Exp}\left[\textsc{Game}_6\left(1^\lambda\right)\right] - \mathsf{Exp}\left[\textsc{Game}_7\left(1^\lambda\right)\right]\right|$ is negligible.

$\textsc{Game}_7\left(1^\lambda\right)$ simulates $\mathbb{E}^{\mathsf{PRC}}_{\mathcal{A},\mathcal{S}}\langle\beta = 1\rangle\left(1^\lambda\right)$ and outputs β'. We conclude that

$$\textsc{Game}_7\left(1^\lambda\right) = \mathsf{Exp}[\beta' \mid \beta = 1]$$

Algorithm 3.5 GAME$_5$ for Ring-GSW

procedure GAME$_5(1^\lambda)$
$\mathbf{A} \xleftarrow{\$} R_q^{m\times 2}$
$t \leftarrow \chi$
$\mathbf{F} \leftarrow \chi^{m\times 2}$
$\mathbf{C}' := \mathbf{A} \cdot t + \mathbf{F}$
$(\mu, \alpha) \leftarrow \mathcal{AF}(1^\lambda, \mathbf{A})$
$\mathbf{C} := \mathbf{C}' + \mu\mathbf{G}$
$\beta' \leftarrow \mathcal{AG}(1^\lambda, \alpha, \mathbf{C})$
return β'
end procedure

Algorithm 3.6 GAME$_6$ for Ring-GSW

procedure GAME$_6(1^\lambda)$
$\mathbf{a} \xleftarrow{\$} R_q^m$
$\mathbf{b} \xleftarrow{\$} R_q^m$
$\mathbf{A} := \begin{bmatrix}\mathbf{b} & \mathbf{a}\end{bmatrix}$
$(\mu, \alpha) \leftarrow \mathcal{AF}(1^\lambda, \mathbf{A})$
$\mathbf{C} \leftarrow \mathsf{Enc}(\mathbf{A}, \mu)$
$\beta' \leftarrow \mathcal{AG}(1^\lambda, \alpha, \mathbf{C})$
return β'
end procedure

Algorithm 3.7 GAME$_7$ for Ring-GSW

procedure GAME$_7(1^\lambda)$
$(\mathbf{A}, \mathbf{s}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(\mu, \alpha) \leftarrow \mathcal{AF}(1^\lambda, \mathbf{A})$
$\beta = 1$
if $\beta = 1$ **then**
$\mathbf{C} \leftarrow \mathsf{Enc}(\mathbf{A}, \mu)$
else
$\mathbf{C} \xleftarrow{\$} R_q^{m\times 2}$
end if
$\beta' \leftarrow \mathcal{AG}(1^\lambda, \alpha, \mathbf{C})$
return β'
end procedure

where β, β' are the corresponding random variables over the probability space of $\mathbb{E}^{\mathrm{PRC}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$. Altogether we proved that $\mathsf{Exp}[\beta' \mid \beta = 0] - \mathsf{Exp}[\beta' \mid \beta = 1]$ is negligible in λ. □

3.2 Homomorphic Operations for Ring-GSW

The Ring-GSW encryption scheme allows to perform homomorphic operations on the ciphertexts.

- $\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2)$: For Ring-GSW-ciphertexts $\mathbf{C}_1, \mathbf{C}_2$, we define $\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2) := \mathbf{G}^{-1}(\mathbf{C}_1)\mathbf{C}_2$. Observe that

$$\begin{aligned}\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2)\mathbf{s} &= \mathbf{G}^{-1}(\mathbf{C}_1)\mathbf{C}_2\mathbf{s} \\ &= \mathbf{G}^{-1}(\mathbf{C}_1)(\mu_2\mathbf{G}\mathbf{s} + \mathsf{noise}(\mathbf{C}_2)) \\ &= \mu_2\mathbf{G}^{-1}(\mathbf{C}_1)\mathbf{G}\mathbf{s} + \mathbf{G}^{-1}(\mathbf{C}_1)\mathsf{noise}(\mathbf{C}_2) \\ &= \mu_2\mathbf{C}_1\mathbf{s} + \mathbf{G}^{-1}(\mathbf{C}_1)\mathsf{noise}(\mathbf{C}_2) \\ &= \mu_2(\mu_1\mathbf{G}\mathbf{s} + \mathsf{noise}(\mathbf{C}_1)) + \mathbf{G}^{-1}(\mathbf{C}_1)\mathsf{noise}(\mathbf{C}_2) \\ &= \mu_1\mu_2\mathbf{G}\mathbf{s} + \mu_2\mathsf{noise}(\mathbf{C}_1) + \mathbf{G}^{-1}(\mathbf{C}_1)\mathsf{noise}(\mathbf{C}_2)\end{aligned}$$

  and therefore

$$\mathsf{noise}(\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2)) = \mu_2\mathsf{noise}(\mathbf{C}_1) + \mathbf{G}^{-1}(\mathbf{C}_1)\mathsf{noise}(\mathbf{C}_2). \tag{3.3}$$

- $\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2)$: For Ring-GSW-ciphertexts $\mathbf{C}_1, \mathbf{C}_2$, we define $\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2) := \mathbf{C}_1 + \mathbf{C}_2$. Observe that

$$\begin{aligned}\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2)\mathbf{s} &= (\mu_1\mathbf{G}\mathbf{s} + \mathsf{noise}(\mathbf{C}_1))) + (\mu_1\mathbf{G}\mathbf{s} + \mathsf{noise}(\mathbf{C}_2)) \\ &= (\mu_1 + \mu_2)\mathbf{G}\mathbf{s} + \mathsf{noise}(\mathbf{C}_1)) + \mathsf{noise}(\mathbf{C}_2))\end{aligned}$$

  and therefore

$$\mathsf{noise}(\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2)) = \mathsf{noise}(\mathbf{C}_1) + \mathsf{noise}(\mathbf{C}_2). \tag{3.4}$$

- $\mathsf{Neg}(\mathbf{C})$: For a Ring-GSW-ciphertext $\mathbf{C}$, we define $\mathsf{Neg}(\mathbf{C}) := -\mathbf{C}$. Observe that

$$\mathsf{Neg}(\mathbf{C})\mathbf{s} = -(\mu\mathbf{G}\mathbf{s} + \mathsf{noise}(\mathbf{C})) = (-\mu)\mathbf{G}\mathbf{s} - \mathsf{noise}(\mathbf{C})$$

  and therefore

$$\mathsf{noise}(\mathsf{Neg}(\mathbf{C})) = -\mathsf{noise}(\mathbf{C}). \tag{3.5}$$

Using these operations, we could define an Eval function and hence extend the Ring-GSW public-key encryption scheme to a levelled FHE scheme. However, we don't formally define the resulting scheme. Instead we will go one step further and construct a *threshold* levelled FHE scheme based on Ring-GSW in Section 3.3.

3.3 Construction of Threshold Levelled Fully Homomorphic Encryption

This section describes our threshold levelled FHE scheme. Asharov et al. [AJW11] already constructed a threshold levelled FHE scheme based on the BGV FHE scheme by Brakerski et al. [BGV11]. Their main tool was the property that BGV key pairs are additively homomorphic. This way key pairs of different parties can be combined into a single shared key pair. Since BGV and Ring-GSW use conceptually equal key pairs, this observation applies to Ring-GSW, too: Let N_{parties} be the number of parties. We require that each party uses the same vector $\mathbf{a}$ in their public key. For $k \in [N_{\mathsf{parties}}]$,

let $(\mathbf{A}_k, \mathbf{s}_k) = \left(\begin{bmatrix}\mathbf{b}_k & \mathbf{a}\end{bmatrix}, (1, -r_k)^T\right)$ be the key pair of party P_k, constructed via $\mathbf{b}_k = \mathbf{a} \cdot r_k + \mathbf{e}_k$. Then $(\mathbf{A}, \mathbf{s})$ defined by

$$\mathbf{A} = \begin{bmatrix} \sum\limits_{k \in [N_{\text{parties}}]} \mathbf{b}_k & \mathbf{a} \end{bmatrix} \quad \mathbf{s} = \left(1, -\sum_{k \in [N_{\text{parties}}]} r_k\right)^T$$

again fulfills the properties of a Ring-GSW key pair; namely that the first component of the secret key is 1 and that

$$\mathbf{As} = \left(\sum_{k \in [N_{\text{parties}}]} \mathbf{b}_k\right) - \mathbf{a} \sum_{k \in [N_{\text{parties}}]} r_k = \sum_{k \in [N_{\text{parties}}]} (\mathbf{b}_k - \mathbf{a} \cdot r_k) = \sum_{k \in [N_{\text{parties}}]} \mathbf{e}_k$$

is small. In the security proof we'll see that it is sufficient if at least one of the $\mathbf{e}_k$ is chosen honestly from χ. In order to apply the leftover hash lemma in the security proof, $\mathbf{a}$ needs to be distributed uniformly at random. It can be taken from the CRS.

This makes a distributed key generation protocol possible: Each party P_k generates its key pair $(\mathbf{A}_k, \mathbf{s}_k)$ and broadcasts the public key $\mathbf{A}_k$. The parties then locally compute the shared public key. Together they hold a full threshold sharing of the secret key.

Let $\mathbf{C}$ be a GSW-ciphertext that decrypts to μ under the secret key $\mathbf{s}$. The parties can collaboratively decrypt $\mathbf{C}$ as follows. With $i := \lfloor \log q \rfloor$, let $\mathbf{C}[i]$ be the i^{th} row of $\mathbf{C}$. Each party P_k computes and broadcasts $\mathsf{pp}_k := C[i, 2] \cdot r_k$. They locally compute the pre-plaintext $\mathsf{pp} := \mathbf{C}[i, 1] - \sum_{k \in [N_{\text{parties}}]} \mathsf{pp}_k$ and proceed as in the Ring-GSW decryption. This is correct because

$$\mathsf{pp} = \mathbf{C}[i, 1] - \sum_{k \in [N_{\text{parties}}]} \mathbf{C}[i, 2] \cdot r_k = \left\langle \mathbf{C}[i], \left(1, -\sum_{k \in [N_{\text{parties}}]} r_k\right)^T \right\rangle = \langle \mathbf{C}[i], \mathbf{s} \rangle$$

just as in the Ring-GSW decryption. However it is not secure, as the pre-plaintext $\mathsf{pp} = \langle \mathbf{C}[i], \mathbf{s} \rangle$ leaks information about $\mathbf{s}$, because $\mathbf{s} = \left(1, \mathbf{C}[i, 2]^{-1}(\mathbf{C}[i, 1] - \mathsf{pp})\right)^T$ (if $\mathbf{C}[i, 2]$ has a multiplicative inverse). Just as Asharov et al. [AJW11], we circumvent this problem by adding large noise to pp, such that we can apply the so-called smudging lemma (Lemma 3.3.1).

Lemma 3.3.1 (Smudging lemma [AJW11])**.** *For security parameter λ, let $B_1 = B_1(\lambda) \in \mathbb{N}$ and $B_2 = B_2(\lambda) \in \mathbb{N}$ such that B_1/B_2 is negligible. Let $e_1 \in [-B_1, B_1[$ be a constant and $e_2 \xleftarrow{\$} [-B_2, B_2[$. Then $\{e_1 + e_2\}_{\lambda \in \mathbb{N}} \equiv_s \{e_2\}_{\lambda \in \mathbb{N}}$.*

For the proof of the smudging lemma we refer to [AJW11]. The idea is to add a large noise element smdg when computing pp, such that pp can not be distinguished from $\mathsf{pp}' := \mathsf{pp} + \mathbf{C}[i, 2] \sum_{k \in [N_{\text{parties}}]} r_k$. Observe that $\mathsf{pp}' = \langle \mathbf{C}[i], \mathbf{s} \rangle + \mathsf{smdg} + \mathbf{C}[i, 2] \sum_{k \in [N_{\text{parties}}]} r_k = \mathbf{C}[i, 1] + \mathsf{smdg}$ does not leak anything about $\mathbf{s}$.

Definition 3.3.1 (Our threshold levelled FHE scheme)**.** Let $\mathcal{K} = (\mathsf{Gen}_{\mathcal{K}}, \mathsf{Com}_{\mathcal{K}})$ be a commitment scheme. For security parameter λ and depth parameter dpth, let $d \in \mathbb{N}$ be a power of two and let $R = \mathbb{Z}[X]/\langle X^d + 1 \rangle$. Let $q = q(\lambda, \mathsf{dpth})$ be an integer modulus. Let $\chi = \chi(\lambda, \mathsf{dpth})$ be a probability distribution over R_q. Set $m = m(\lambda, \mathsf{dpth}) := 2(\lfloor \log q \rfloor + 1)$. Let $B_{\mathsf{smdg}} \in \mathbb{N}$.

Furthermore, let C be the class of 1-bounded arithmetic circuits. In the common random string model, we define the threshold C-evaluation scheme $\mathsf{TFHE}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},N_{\mathsf{parties}}} = (X, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ for N_{parties} parties as follows.

- $X = \left(X_{\mathsf{pk}}\right)_{\mathsf{pk} \in K_S^{\mathsf{pub}}}$: For every public key $\mathsf{pk} \in K_S^{\mathsf{pub}}$ the plaintext space is $X_{\mathsf{pk}} = \{0,1\}$.
- $\mathsf{Gen}\left(1^\lambda,\ \mathsf{dpth}\right)$: Each party P_k — $k \in [N_{\mathsf{parties}}]$ — does the following. Using random coins from the CRS, compute $\mathsf{params}_{\mathcal{K}} \leftarrow \mathsf{Gen}_{\mathcal{K}}\left(1^\lambda\right)$. Take $\mathbf{a}$ from the CRS, sample $r_k \leftarrow \chi$; $\mathbf{e}_k \leftarrow \chi^m$ and set $\mathbf{b}_k := \mathbf{a} \cdot r_k + \mathbf{e}_k$. Sample the randomness $\alpha_k \xleftarrow{\$} \{0,1\}^t$ (where t is large enaugh) in order to commit $c_k := \mathsf{Com}_{\mathcal{K}}^{\alpha_k}\left(\mathsf{params}_{\mathcal{K}}, \mathbf{b}_k\right)$ and broadcast c_k. Upon receiving c_j from all other parties P_j, $j \in [N_{\mathsf{parties}}]$, broadcast $(\mathbf{b}_k, \alpha_k)$. If there exists $j \in [N_{\mathsf{parties}}]$ such that $c_j \neq \mathsf{Com}_{\mathcal{K}}^{\alpha_j}\left(\mathsf{params}_{\mathcal{K}}, \mathbf{b}_j\right)$, then abort. Locally compute $\mathbf{A} := \left[\sum_{k \in [N_{\mathsf{parties}}]} \mathbf{b}_k \quad \mathbf{a}\right]$ and output $\left(\mathsf{pk} = \mathbf{A}, \mathsf{sk} = (r_k)_{k \in [N_{\mathsf{parties}}]}\right)$.
- $\mathsf{Enc}\left(\mathsf{pk} = \mathbf{A} : R_q^{m \times 2},\ \mu : \{0,1\}\right)$: Sample $t \leftarrow \chi$; $\mathbf{F} \leftarrow \chi^{m \times 2}$ and return $\mathbf{C} := t\mathbf{A} + \mathbf{F} + \mu\mathbf{G}$. (This is the same as Ring-GSW encryption.)
- $\mathsf{Eval}\left(\mathsf{evk} = \bot,\ f : C,\ \left(\mathbf{C}_1, \ldots, \mathbf{C}_{N_{\mathsf{inputs}}}\right) : \left(R_q^{m \times 2}\right)^{N_{\mathsf{inputs}}}\right)$: Note that, as per Definition 2.6.1, N_{inputs} is the number of input gates of f. For each $i \in [N_{\mathsf{inputs}}]$, let in_i be the i^{th} input gate and set $\mathbf{C}_{\mathsf{in}_i} := \mathbf{C}_i$. Process all other gates in topological order as follows. When processing an
  - addition gate g with incoming wires from g_1 and g_2, set $\mathbf{C}_g := \mathsf{Add}\left(\mathbf{C}_{g_1}, \mathbf{C}_{g_2}\right)$;
  - multiplication gate g with incoming wires from g_1 and g_2, set $\mathbf{C}_g := \mathsf{Mult}\left(\mathbf{C}_{g_1}, \mathbf{C}_{g_2}\right)$;
  - negation gate g with incoming wire from g_1, set $\mathbf{C}_g := \mathsf{Neg}\left(\mathbf{C}_{g_1}\right)$;
  - output gate g with incoming wire from g_1, set $\mathbf{C}_g := \mathbf{C}_{g_1}$.

  For each $i \in [N_{\mathsf{outputs}}]$, let out_i be the i^{th} output gate. Return $\left(\mathbf{C}_{\mathsf{out}_1}, \ldots, \mathbf{C}_{\mathsf{out}_{N_{\mathsf{outputs}}}}\right)$.
- $\mathsf{Dec}\left(\mathsf{sk} = (r_k)_{k \in [N_{\mathsf{parties}}]} : R_q^{N_{\mathsf{parties}}}, \mathbf{C} : R_q^{m \times 2}\right)$: With $i := \lfloor \log q \rfloor$, let $\mathbf{C}[i]$ be the i^{th} row of $\mathbf{C}$. Each party P_k samples $\mathsf{smdg}_k \xleftarrow{\$} R_{[-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}[}$ and broadcasts $\mathsf{pp}_k := \mathbf{C}[i,2] \cdot r_k + \mathsf{smdg}_k$. They locally compute $\mathsf{pp} := \mathbf{C}[i,1] - \sum_{k \in [N_{\mathsf{parties}}]} \mathsf{pp}_k$ and proceed as in Ring-GSW decryption.

Correct decryption can only be guaranteed if the noise does not grow too large. One can choose the parameters $d(\lambda, \mathsf{dpth})$, $q(\lambda, \mathsf{dpth})$ and $\chi(\lambda, \mathsf{dpth})$ such that correct evaluation is guaranteed for any 1-bounded arithmetic circuit f of depth at most dpth, hence obtaining a threshold levelled FHE scheme (Definition 2.6.2). As a step towards choosing the right parameters, we prove the following more low-level lemma. Compared to Ring-GSW public-key encryption, the smudging requires us to lower the bound on the noise the ciphertext is allowed to contain in order to still maintain correct decryption.

Lemma 3.3.2. *Let* $\mathbf{s} := \left(1, -\sum_{k \in [N_{\mathsf{parties}}]} r_k\right)^T$. *If*

$$\left\|\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C})\right\|_\infty + N_{\mathsf{parties}} B_{\mathsf{smdg}} < 2^{\lfloor \log q \rfloor - 2}$$

then $\mathsf{TFHE}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},N_{\mathsf{parties}}}$ *decryption is correct, i.e.* $\mathsf{Dec}(\mathsf{sk}, C) = \mu$.

Proof. Let $i = \lfloor \log q \rfloor$. First observe that

$$\begin{aligned}
\mathsf{pp} &= \mathbf{C}[i,1] - \sum_{k \in [N_{\mathsf{parties}}]} \mathsf{pp}_k \\
&= \mathbf{C}[i,1] - \sum_{k \in [N_{\mathsf{parties}}]} (\mathbf{C}[i,2] \cdot r_k + \mathsf{smdg}_k) \\
&= \mathbf{C}[i,1] - \sum_{k \in [N_{\mathsf{parties}}]} (\mathbf{C}[i,2] \cdot r_k) - \sum_{k \in [N_{\mathsf{parties}}]} \mathsf{smdg}_k \\
&= \left\langle \mathbf{C}[i], \left(1, \sum_{k \in [N_{\mathsf{parties}}]} r_k\right)\right\rangle - \sum_{k \in [N_{\mathsf{parties}}]} \mathsf{smdg}_k \\
&= \langle \mathbf{C}[i], \mathbf{s} \rangle - \sum_{k \in [N_{\mathsf{parties}}]} \mathsf{smdg}_k .
\end{aligned}$$

It follows that

$$\begin{aligned}
\left|\left|\mathsf{pp} - \mu 2^{i-1}\right|\right|_\infty &\leq \left|\left|\langle \mathbf{C}[i], \mathbf{s} \rangle - \mu 2^{i-1}\right|\right|_\infty + \sum_{k \in [N_{\mathsf{parties}}]} ||\mathsf{smdg}_k||_\infty \\
&\leq ||\mathbf{Cs} - \mu \mathbf{Gs}||_\infty + N_{\mathsf{parties}} B_{\mathsf{smdg}} \\
&= \left|\left|\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C})\right|\right|_\infty + N_{\mathsf{parties}} B_{\mathsf{smdg}} \\
&< 2^{i-2}.
\end{aligned}$$

We can proceed analogously to the proof of Lemma 3.1.1. □

According to this lemma, we can choose a feasible q if we have an upper bound on the noise of ciphertexts that shall be decrypted. Those are the ciphertexts that are returned by Eval.

Lemma 3.3.3. *Let* $\mathbf{s} := \left(1, -\sum_{k \in [N_{\mathsf{parties}}]} r_k\right)^T$. *Let* $B_\chi \in \mathbb{N}$ *such that* χ *is* B_χ*-bounded. Let* f *be an arithmetic circuit of depth at most* dpth *and let* N_{inputs} *be the number of input gates of* f. *Without loss of generality,* f *has only a single output. For* $i \in [N_{\mathsf{inputs}}]$, *let* $\mu_i \in \{0,1\}$ *and* $\mathbf{C}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_i)$. *Set* $\mu := f(\mu_1, \ldots, \mu_{N_{\mathsf{inputs}}})$ *and* $\mathbf{C} \leftarrow \mathsf{Eval}\left(\perp, f, \left(\mathbf{C}_1, \ldots, \mathbf{C}_{N_{\mathsf{inputs}}}\right)\right)$. *Then*

$$\left|\left|\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C})\right|\right|_\infty \leq \left(N_{\mathsf{parties}} + 1\right) B_\chi^2 \cdot (m+1)^{\mathsf{dpth}}.$$

Proof. Recall that in the computation of $\mathbf{C} \leftarrow \mathsf{Eval}\left(\perp, f, \left(\mathbf{C}_1, \ldots, \mathbf{C}_{N_{\mathsf{inputs}}}\right)\right)$ each gate g of f gets an assigned ciphertext $\mathbf{C}_g$. Furthermore, let μ_g be the "correct" plaintext for that gate; that is, the value of that gate when (non-homomorphically) evaluating $f(\mu_1, \ldots, \mu_{N_{\mathsf{inputs}}})$. We define

$$\mathsf{layer}(g) = \begin{cases} 0 & \text{if } g \text{ is an input gate,} \\ 1 + \mathsf{layer}(g_1) & \text{if } g \text{ has one incoming wire from } g_1, \\ 1 + \max\{\mathsf{layer}(g_1), \mathsf{layer}(g_2)\} & \text{if } g \text{ has two incoming wires from } g_1, g_2. \end{cases}$$

By induction over $l \in \{0, \ldots, \mathsf{dpth}\}$ we first prove that for each non-output gate g of f with $\mathsf{layer}(g) = l$ it holds true that $\left|\left|\mathsf{noise}_{\mathbf{s},\mu_g}\left(\mathbf{C}_g\right)\right|\right|_\infty \leq \left(N_{\mathsf{parties}} + 1\right) B_\chi^2 \cdot (m+1)^l$.

- Let $l = 0$. Then $\mathbf{C}_g$ is a fresh ciphertext with

$$\begin{aligned}\mathsf{noise}_{\mathbf{s},\mu_g}(\mathbf{C}_g) &= (\mathbf{C}_g - \mu_g\mathbf{G})\mathbf{s} = (t\mathbf{A} + \mathbf{F})\mathbf{s} = t\mathbf{A}\mathbf{s} = \mathbf{F}\mathbf{s} \\ &= t\Big[\textstyle\sum_{k\in[N_{\mathsf{parties}}]} \mathbf{b}_k \quad \mathbf{a}\Big]\mathbf{s} + \mathbf{F}\mathbf{s} \\ &= t\Big[\textstyle\sum_{k\in[N_{\mathsf{parties}}]} (\mathbf{a}\cdot r_k + \mathbf{e}_k) \quad \mathbf{a}\Big]\begin{pmatrix} 1 \\ -\sum_{k\in[N_{\mathsf{parties}}]} r_k \end{pmatrix} + \mathbf{F}\mathbf{s} \\ &= t\sum_{k\in[N_{\mathsf{parties}}]} \mathbf{e}_k + \mathbf{F}\mathbf{s} = \sum_{k\in[N_{\mathsf{parties}}]} t\mathbf{e}_k + \mathbf{F}\mathbf{s}.\end{aligned}$$

  All of the polynomials in the latter term are sampled from the error distribution χ, and hence $\big\|\mathsf{noise}_{\mathbf{s},\mu_g}(\mathbf{C}_g)\big\|_\infty \leq N_{\mathsf{parties}}B_\chi^2 + B_\chi^2 = \big(N_{\mathsf{parties}}+1\big)B_\chi^2$.

- Let $l \in \{1,\dots,\mathsf{dpth}\}$ where g is a multiplication gate with incoming wires from g_1 and g_2. By Equation (3.3) we have $\mathbf{C}_g = \mu_{g_2}\mathsf{noise}\big(\mathbf{C}_{g_1}\big) + \mathbf{G}^{-1}\big(\mathbf{C}_{g_1}\big)\mathsf{noise}\big(\mathbf{C}_{g_2}\big)$. Using $\mu_{g_2} \in \{0,1\}$, $\mathbf{G}^{-1}\big(\mathbf{C}_{g_1}\big) \in R^{m\times m}_{[-1,1]}$ by Equation (3.1) and

$$\big\|\mathsf{noise}\big(\mathbf{C}_{g_1}\big)\big\|_\infty, \big\|\mathsf{noise}\big(\mathbf{C}_{g_2}\big)\big\|_\infty \leq \big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l-1}$$

  by induction, we obtain

$$\begin{aligned}\big\|\mathbf{C}_g\big\|_\infty &\leq \big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l-1} + m\big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l-1} \\ &= \big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l}.\end{aligned}$$

- Let $l \in \{1,\dots,\mathsf{dpth}\}$ where g is an addition gate with incoming wires from g_1 and g_2. By Equation (3.4) we have $\mathbf{C}_g = \mathsf{noise}\big(\mathbf{C}_{g_1}\big) + \mathsf{noise}\big(\mathbf{C}_{g_2}\big)$ and hence we obtain

$$\big\|\mathbf{C}_g\big\|_\infty \leq 2\big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l-1} < \big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l}.$$

- Let $l \in \{1,\dots,\mathsf{dpth}\}$ where g is a negation gate with incoming wire from g_1. By Equation (3.5) we have $\mathbf{C}_g = -\mathsf{noise}\big(\mathbf{C}_{g_1}\big)$ and hence we obtain

$$\big\|\mathbf{C}_g\big\|_\infty = \big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l-1} < \big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l}.$$

At the end of the computation, the output $\mathbf{C}$ is taken from an output gate (which doesn't count towards the depth). Let g be the predecessor of that output gate. Then $\mathbf{C} = \mathbf{C}_g$. Since the depth of f is at most dpth, it follows $l := \mathsf{layer}(g) \leq \mathsf{dpth}$. We conclude

$$\big\|\mathsf{noise}_{\mathbf{s},\mu}(\mathbf{C})\big\|_\infty = \big\|\mathsf{noise}_{\mathbf{s},\mu_g}(\mathbf{C}_g)\big\|_\infty \leq N_{\mathsf{parties}}\big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{l}.$$

□

Theorem 3.3.1. *Let $B_\chi \in \mathbb{N}$ such that χ is B_χ-bounded. If for all $\lambda, \mathsf{dpth} \in \mathbb{N}$ it holds true that*

$$q > 8\Big(\big(N_{\mathsf{parties}}+1\big)B_\chi^2\cdot(m+1)^{\mathsf{dpth}} + N_{\mathsf{parties}}B_{\mathsf{smdg}}\Big) \tag{3.6}$$

then $\mathsf{TFHE}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},N_{\mathsf{parties}}}$ is a threshold levelled FHE scheme (as defined in Definition 2.6.2).

Proof. Let $\mathsf{dpth} \in \mathbb{N}$. We have to prove that $\mathcal{S}' := (X, \mathsf{Gen}(\cdot, \mathsf{dpth}), \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is a threshold C_{dpth}-evaluation scheme (as defined in Definition 2.6.1) where C_{dpth} is the class of 1-bounded arithmetic circuits of depth at most dpth. To do so it is left to prove that $\mathcal{S}'$ fulfills the correctness property. That is, for all $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \in K_{\mathcal{S}'}$, $f \in C_{\mathsf{dpth}}$ and $\mu = (\mu_i)_{i \in [N_{\mathsf{inputs}}]} \in X_{\mathsf{pk}}^{N_{\mathsf{inputs}}}$ it shall hold true that

$$\Pr\left[\mathsf{Dec}\left(\mathsf{sk}, \mathsf{Eval}\left(\mathsf{evk}, f, \mathsf{Enc}(\mathsf{pk}, \mu_i)_{i \in [N_{\mathsf{inputs}}]}\right)\right) = f(\mu)\right] = 1.$$

Note that $\mathsf{evk} = \perp$ in our case. Let $\mathbf{C} \leftarrow \mathsf{Eval}(\perp, f, \mathsf{Enc}(\mathsf{pk}, \mu))$. By Lemma 3.3.3 we have that $\left\|\mathsf{noise}_{\mathbf{s}, f(\mu)}(\mathbf{C})\right\|_\infty \leq \left(N_{\mathsf{parties}} + 1\right) B_\chi^2 \cdot (m+1)^{\mathsf{dpth}}$ where $\mathbf{s} := \left(1, -\sum_{k \in [N_{\mathsf{parties}}]} r_k\right)^T$. Consequently

$$\begin{aligned}\left\|\mathsf{noise}_{\mathbf{s}, f(\mu)}(\mathbf{C})\right\|_\infty + N_{\mathsf{parties}} B_{\mathsf{smdg}} &\leq \left(N_{\mathsf{parties}} + 1\right) B_\chi^2 \cdot (m+1)^{\mathsf{dpth}} + N_{\mathsf{parties}} B_{\mathsf{smdg}} \\ &< q/8 = 2^{\log q - 3} < 2^{\lfloor \log q \rfloor - 2}.\end{aligned}$$

That is the premise of Lemma 3.3.2 which yields that $\mathsf{Dec}(\mathsf{sk}, \mathbf{C}) = f(\mu)$. □

In Equation (3.6) the term $\left(N_{\mathsf{parties}} + 1\right) B_\chi^2 \cdot (m+1)^{\mathsf{dpth}}$ bounds the noise which the resulting ciphertext after evaluation is allowed to have. As we saw in the proof, the term is chosen because every ciphertext that results from an evaluation of depth at most dpth satisfies that bound. However, there might be a better bound for certain classes of arithmetic circuits and in those cases we might be able to evaluate circuits of a greater depth, i.e. a depth dpth that does not satisfy Equation (3.6). Specifically, we can evaluate a circuit if we can prove that the resulting ciphertext's noise has an upper bound $B_{\mathsf{noise}} < q/8 - N_{\mathsf{parties}} B_{\mathsf{smdg}}$.

3.4 Our Passively Secure MPC Protocol

In this section we define our passively secure MPC protocol and prove that it provides correctness and computational privacy in the presence of passive adversaries.

Our passively secure MPC protocol uses a commitment scheme $\mathcal{K}$. Later we will prove the protocol's security under certain requirements. One of those requirements will be that $\mathcal{K}$ is perfectly hiding and computationally binding.

Definition 3.4.1 (Our passively secure MPC protocol)**.** Let $\mathcal{K} = (\mathsf{Gen}_{\mathcal{K}}, \mathsf{Com}_{\mathcal{K}})$ be a commitment scheme. Let f be a 1-bounded arithmetic circuit over R_q. Let N_{inputs} be the number of input gates and dpth be the depth of f. For security parameter λ and depth parameter dpth as specified above, let $d, R, q, \chi, B_{\mathsf{smdg}}$ be as in Definition 3.3.1. Furthermore, there are N_{parties} parties $P_1, \ldots, P_{N_{\mathsf{parties}}}$ and $\phi : [N_{\mathsf{inputs}}] \to [N_{\mathsf{parties}}]$ assigns each input gate to a party. For each party P_k, the protocol input has the form $\mathbf{x}_k := (\mu_i)_{i \in \phi^{-1}(k)}$. In the common random string model, our passively secure MPC protocol $\mathsf{PassiveMPC}_{\mathcal{K}, d, q, \chi, B_{\mathsf{smdg}}, f, \phi}\left(1^\lambda, (\mathbf{x}_k)_{k \in [N_{\mathsf{parties}}]}\right)$ is defined as follows, utilizing $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}) = \mathsf{TFHE}_{\mathcal{K}, d, q, \chi, B_{\mathsf{smdg}}, N_{\mathsf{parties}}}$ from Definition 3.3.1.

Round I. The parties run the first round of the key generation protocol $\mathsf{Gen}\left(1^\lambda, \mathsf{dpth}\right)$.

Round II. The parties run the second round of the key generation protocol, such that every party obtains the public key pk and their share of the secret key sk.

Round III. For each $i \in [N_{\mathsf{inputs}}]$: Let in_i be the i^{th} input gate. Party $P_{\phi(i)}$ computes $\mathbf{C}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_i)$ and broadcasts $(i, \mathbf{C}_i)$.

Round IV. The parties compute $\left(\mathbf{C}_{\mathsf{out}_1}, \ldots, \mathbf{C}_{\mathsf{out}_{N_{\mathsf{outputs}}}}\right) \leftarrow \mathsf{Eval}\left(\perp, f, \left(\mathbf{C}_{\mathsf{in}_1}, \ldots, \mathbf{C}_{\mathsf{in}_{N_{\mathsf{inputs}}}}\right)\right)$. In parallel for each $i \in \left[N_{\mathsf{outputs}}\right]$ they perform the one-round decryption protocol $\mu_i' \leftarrow \mathsf{Dec}\left(\mathsf{sk}, \mathbf{C}_{\mathsf{out}_i}\right)$. They return $\left(\mu_1', \ldots, \mu_{N_{\mathsf{outputs}}}'\right)$.

Theorem 3.4.1. *Let $\mathcal{P} = \mathsf{PassiveMPC}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},f,\phi}$ and $B_\chi \in \mathbb{N}$ such that χ is B_χ-bounded. If for all $\lambda \in \mathbb{N}$, for all circuit inputs $\mu = (\mu_i)_{i\in[N_{\mathsf{inputs}}]}$ and for all output ciphertexts $\mathbf{C}_{\mathsf{out}_i}$, $i \in \left[N_{\mathsf{outputs}}\right]$, it holds true that*

$$\mathsf{Pr}\left[\left\|\mathsf{noise}_{\mathbf{s},f(\mu)}\mathbf{C}_{\mathsf{out}_i}\right\|_\infty < q/8 - N_{\mathsf{parties}}B_{\mathsf{smdg}}\right] = 1$$

over the probability space of $\mathcal{P}\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\mathsf{parties}}]}\right)$, then $\mathcal{P}$ provides correctness in the presence of passive adversaries (as defined in Definition 2.6.5) with respect to $(\mathbf{x}_k)_{k\in[N_{\mathsf{parties}}]} \mapsto f(\mu)$.

Proof. Let $\mathcal{P} = \mathsf{PassiveMPC}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},f,\phi}$. We have to prove that for all runs of the protocol $y \leftarrow \mathcal{P}\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\mathsf{parties}}]}\right)$ (with honest parties) it holds true that $y \in \{f(\mu), \perp\}$ Because the parties never abort, we will prove that $y = f(\mu)$. Let $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$ be the output of $\mathsf{Gen}\left(1^\lambda, \mathsf{dpth}\right)$ in the considered run. By Theorem 3.3.1, $\mathsf{TFHE}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},N_{\mathsf{parties}}}$ is a threshold levelled FHE scheme (Definition 2.6.2), so in particular, $(X, \mathsf{Gen}(\cdot, \mathsf{dpth}), \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is a C_{dpth}-evaluation scheme (Definition 2.6.1) where C_{dpth} is the class of 1-bounded arithmetic circuits of depth at most dpth. Since dpth is the depth of f and thus $f \in C_{\mathsf{dpth}}$, we can use the correctness property of Definition 2.6.1 which states that

$$\mathsf{Pr}\left[\mathsf{Dec}\left(\mathsf{sk}, \mathsf{Eval}\left(\mathsf{evk}, (\mathsf{Enc}(\mathsf{pk}, \mu_i))_{i\in[N_{\mathsf{inputs}}]}\right)\right) = f(\mu)\right] = 1.$$

As y is drawn from $\mathsf{Dec}\left(\mathsf{sk}, \mathsf{Eval}\left(\mathsf{evk}, f, (\mathsf{Enc}(\mathsf{pk}, \mu_i))_{i\in[N_{\mathsf{inputs}}]}\right)\right)$, this concludes the proof. □

Theorem 3.4.2. *If*

- *$\mathcal{K}$ is perfectly hiding and computationally binding,*
- *the $\mathsf{RLWE}_{d,q,\chi}$ assumption holds true and*
- *B_{smdg}/B_χ is negligible,*

then $\mathsf{PassiveMPC}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},f,\phi}$ provides computational privacy in the presence of passive adversaries (as defined in Definition 2.6.6).

Proof. Let $\mathcal{P} := \mathsf{PassiveMPC}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},f,\phi}$. Consider a tuple of ppt adversaries $\left(P_k^*\right)_{k\in K}$ for some $K \subseteq \left[N_{\mathsf{parties}}\right]$. If $K = \left[N_{\mathsf{parties}}\right]$, i.e. every party is an adversary, then we can trivially use the simulator $\mathcal{S}(1^\lambda, x, y) := \mathcal{P}^*(1^\lambda, x)$. Therefore from now on we assume that $K \neq \left[N_{\mathsf{parties}}\right]$. Without loss of generality let $1 \notin K$. Define the simulator $\mathcal{S}$ as in Algorithm 3.8. We give a

sequence of games such that — for all input vectors $x = \left(\mathbf{x}_1, \ldots, \mathbf{x}_{N_{\text{parties}}}\right) \in \left(\{0,1\}^*\right)^{N_{\text{parties}}}$ — the first game equals $\mathcal{S}\left(1^\lambda, (\mathbf{x}_k)_{k\in K}, y\right)$, the last game equals $\pi_K\left(\mathcal{P}^*\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}\right)\right)$ and consecutive games are computationally indistinguishable. Let $\text{GAME}_1 = \mathcal{S}$.

Algorithm 3.8 Simulator for PassiveMPC

procedure $\mathcal{S}(1^\lambda, (\mathbf{x}_k)_{k\in K}, y = \left(\mu'_j\right)_{j\in[N_{\text{outputs}}]})$
  for $k \in [N_{\text{parties}}]$ **do**
    $\tilde{\mathbf{x}}_k := \begin{cases} \mathbf{x}_k & k \in K \\ 0^{|\phi^{-1}(k)|} & \text{else} \end{cases}$
  end for
  Simulate the first three rounds of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$.
  $i := \lfloor \log q \rfloor$
  for $j \in [N_{\text{outputs}}]$ **do**
    $\mathsf{smdg} \xleftarrow{\$} R_{[-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}[}$
    $\mathsf{pp}_1^{(j)} := -\mu'_j 2^{i-1} + \mathsf{smdg} + \mathbf{C}_{\text{out}_j}[i,1] - \mathbf{C}_{\text{out}_j}[i,2] \cdot \sum_{k=2}^{N_{\text{parties}}} r_k$
    Modify P_1 such that in the subprotocol $\mathsf{Dec}\left(\mathsf{sk}, \mathbf{C}_{\text{out}_j}\right)$ it sends $\mathsf{pp}_1^{(j)}$ instead of pp_1.
  end for
  Simulate the forth round of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$, obtaining output z.
  return $\pi_K(z)$
end procedure

GAME_2 is presented in abbreviated form in Algorithm 3.9. It is the same as GAME_1 expect that $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}_1$ are precomputed right at the beginning. This is a purely syntactic change.

Algorithm 3.9 GAME_2 for PassiveMPC

procedure $\text{GAME}_2(1^\lambda, (\mathbf{x}_k)_{k\in K}, y = \left(\mu'_j\right)_{j\in[N_{\text{outputs}}]})$
  $\tilde{\mathbf{a}} \xleftarrow{\$} R_q^m$
  $\tilde{r}_1 \leftarrow \chi$
  $\tilde{\mathbf{e}}_1 \leftarrow \chi^m$
  $\tilde{\mathbf{b}}_1 := \mathbf{a} \cdot r_1 + \mathbf{e}_1$
  $\mathcal{P}^* := \mathcal{P}^*\left\langle \mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b}_1 = \tilde{\mathbf{b}}_1 \right\rangle$
  for $k \in [N_{\text{parties}}]$ **do**
    $\tilde{\mathbf{x}}_k := \begin{cases} \mathbf{x}_k & k \in K \\ 0^{|\phi^{-1}(k)|} & \text{else} \end{cases}$
  end for
  Simulate the first three rounds of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  ⋮
end procedure

Observe how GAME_2 is similar to the $\beta = 1$ case of an Ring-LWE security game. We obtain GAME_3 (see Algorithm 3.10) from GAME_2 by replacing $r_1 \leftarrow \chi$; $\mathbf{e}_1 \leftarrow \chi^m$; $\mathbf{b}_1 := \mathbf{a} \cdot r_1 + \mathbf{e}_1$ by $\mathbf{b}_1 \xleftarrow{\$} R_q^m$. Note that this replacement is possible because r_1 and $\mathbf{e}_1$ aren't used anywhere else in the simulation. Let U be a distinguisher for $G_1 := \{\text{GAME}_1(1^\lambda, (\mathbf{x}_k)_{k\in K}, y)\}_{\lambda\in\mathbb{N}}$ and $G_2 := \{\text{GAME}_2(1^\lambda, (\mathbf{x}_k)_{k\in K}, y)\}_{\lambda\in\mathbb{N}}$. We define the ppt adversary $\mathcal{A}$ as follows. On input $(1^\lambda, \tilde{\mathbf{a}}, \tilde{\mathbf{b}})$, $\mathcal{A}$ simulates $\text{GAME}_2(1^\lambda, (\mathbf{x}_k)_{k\in K}, y)$ from line $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b}_1 = \tilde{\mathbf{b}}_1\rangle$ onwards and runs U on the simulation output. We observe that $\mathbb{E}^{\text{dist}}_{U,G_1,G_2}(1^\lambda) \equiv \mathbb{E}^{\text{RLWE}}_{\mathcal{A},d,q,\chi,m}(1^\lambda)$ and therefore

$$\begin{aligned}
\left|\text{Adv}^{\text{dist}}_{U,G_1,G_2}(\lambda)\right| &= \left|1 - 2\text{Exp}\left[\mathbb{E}^{\text{dist}}_{U,G_1,G_2}\left(1^\lambda\right)\right]\right| \\
&= \left|1 - 2\text{Exp}\left[\mathbb{E}^{\text{RLWE}}_{\mathcal{A},d,q,\chi,m}\left(1^\lambda\right)\right]\right| \\
&= \left|\text{Adv}^{\text{RLWE}}_{\mathcal{A},d,q,\chi,m}(\lambda)\right| \\
&= \text{negl}(\lambda)
\end{aligned}$$

where the latter is negligible by the $\text{RLWE}_{d,q,\chi}$ assumption.

Algorithm 3.10 GAME_3 for PassiveMPC

procedure $\text{GAME}_3(1^\lambda, (\mathbf{x}_k)_{k\in K}, y = \left(\mu'_j\right)_{j\in[N_{\text{outputs}}]})$
  $\tilde{\mathbf{a}} \xleftarrow{\$} R_q^m$
  $\tilde{\mathbf{b}}_1 \xleftarrow{\$} R_q^m$
  $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b}_1 = \tilde{\mathbf{b}}_1\rangle$
  for $k \in [N_{\text{parties}}]$ **do**

$$\tilde{\mathbf{x}}_k := \begin{cases} \mathbf{x}_k & k \in K \\ 0^{|\phi^{-1}(k)|} & \text{else} \end{cases}$$

  end for
  Simulate the first three rounds of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  ⋮
end procedure

GAME_4 (see Algorithm 3.11) is the same as GAME_3 except that the ciphertexts of the honest parties are precomputed before the third round of the simulation. This is again a purely syntactic change.

Observe that in GAME_4 we have $\mathbf{A} := \left[\sum_{k\in[N_{\text{parties}}]} \mathbf{b}_k \quad \mathbf{a}\right]$ with $\mathbf{a}, \mathbf{b}_1 \xleftarrow{\$} R_q^m$. Because of the perfectly hiding property of the commitment scheme, the initially committed values of $\mathbf{b}_k$ for $k \neq 1$ (Note that even the adversaries know such values because they are only passive adversaries.) are independent of $\mathbf{b}_1$. Furthermore, because of the computational binding property of the commitment scheme, there is a negligible probability that the opened values of $\mathbf{b}_k$ do not equal the initialy committed values. When excluding this error event of negligible probability, $\mathbf{b} := \sum_{k\in[N_{\text{parties}}]} \mathbf{b}_k$ is consequently still distributed uniformly at random. It follows that $\mathbf{A}$ is distributed uniformly at random over $R_q^{m\times 2}$. Therefore we can proceed analogous to the proof of Theorem 3.1.1 and replace $\text{Enc}(\mathbf{A}, 0)$ by uniformly chosen ciphertexts. That way we obtain GAME_5 as presented in Algorithm 3.12 which is computationally indistinguishable from GAME_4.

We obtain GAME_6 (see Algorithm 3.13) by performing the previous step in reverse, replacing the uniformly chosen ciphertexts in GAME_5 by encryptions of the "correct" input values.

Algorithm 3.11 GAME$_4$ for PassiveMPC

procedure GAME$_4$($1^\lambda, (\mathbf{x}_k)_{k\in K}, y = \left(\mu'_j\right)_{j\in[N_{\text{outputs}}]}$)
  $\tilde{\mathbf{a}} \xleftarrow{\$} R_q^m$
  $\tilde{\mathbf{b}}_1 \xleftarrow{\$} R_q^m$
  $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b}_1 = \tilde{\mathbf{b}}_1 \rangle$
  for $k \in [N_{\text{parties}}]$ **do**
    $$\tilde{\mathbf{x}}_k := \begin{cases} \mathbf{x}_k & k \in K \\ 0^{|\phi^{-1}(k)|} & \text{else} \end{cases}$$
  end for
  Simulate the first two rounds of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  for $j \in [N_{\text{inputs}}]$ **do**
    if $\phi(j) \notin K$ **then**
      $\tilde{\mathbf{C}} \leftarrow \text{Enc}(\mathbf{A}, 0)$
      $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{C}_{\text{in}_j} = \tilde{\mathbf{C}} \rangle$
    end if
  end for
  Simulate the third round of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  ⋮
end procedure

Algorithm 3.12 GAME$_5$ for PassiveMPC

procedure GAME$_5$($1^\lambda, (\mathbf{x}_k)_{k\in K}, y = \left(\mu'_j\right)_{j\in[N_{\text{outputs}}]}$)
  $\tilde{\mathbf{a}} \xleftarrow{\$} R_q^m$
  $\tilde{\mathbf{b}}_1 \xleftarrow{\$} R_q^m$
  $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b}_1 = \tilde{\mathbf{b}}_1 \rangle$
  for $k \in [N_{\text{parties}}]$ **do**
    $$\tilde{\mathbf{x}}_k := \begin{cases} \mathbf{x}_k & k \in K \\ 0^{|\phi^{-1}(k)|} & \text{else} \end{cases}$$
  end for
  Simulate the first two rounds of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  for $j \in [N_{\text{inputs}}]$ **do**
    if $\phi(j) \notin K$ **then**
      $\tilde{\mathbf{C}} \xleftarrow{\$} R_q^{m\times 2}$
      $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{C}_{\text{in}_j} = \tilde{\mathbf{C}} \rangle$
    end if
  end for
  Simulate the third round of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  ⋮
end procedure

Algorithm 3.13 GAME$_6$ for PassiveMPC

procedure GAME$_6$($1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}, y = \left(\mu'_j\right)_{j\in[N_{\text{outputs}}]}$)
  $\tilde{\mathbf{a}} \xleftarrow{\$} R_q^m$
  $\tilde{\mathbf{b}}_1 \xleftarrow{\$} R_q^m$
  $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b}_1 = \tilde{\mathbf{b}}_1 \rangle$
  for $k \in [N_{\text{parties}}]$ **do**
    $$\tilde{\mathbf{x}}_k := \begin{cases} \mathbf{x}_k & k \in K \\ 0^{|\phi^{-1}(k)|} & \text{else} \end{cases}$$
  end for
  Simulate the first two rounds of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  for $j \in [N_{\text{inputs}}]$ **do**
    if $\phi(j) \notin K$ **then**
      $\tilde{\mathbf{C}} \leftarrow \text{Enc}(\mathbf{A}, \mu_j)$
      $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{C}_{\text{in}_j} = \tilde{\mathbf{C}} \rangle$
    end if
  end for
  Simulate the third round of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$
  ⋮
end procedure

Observe that GAME$_6$ runs a simulation of the first three rounds of the protocol on the "correct" input values $(\mathbf{x}_k)_{k\in[N_{\text{parties}}]}$, up to some syntactic differences. We obtain GAME$_7$ (see Algorithm 3.14 by eliminating those syntactic differences, which is a purely syntactic change.

Algorithm 3.14 GAME$_7$ for PassiveMPC

procedure GAME$_7$($1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}, y = \left(\mu'_j\right)_{j\in[N_{\text{outputs}}]}$)
  $\tilde{\mathbf{a}} \xleftarrow{\$} R_q^m$
  $\tilde{\mathbf{b}}_1 \xleftarrow{\$} R_q^m$
  $\mathcal{P}^* := \mathcal{P}^*\langle \mathbf{a} = \tilde{\mathbf{a}}, \mathbf{b}_1 = \tilde{\mathbf{b}}_1 \rangle$
  Simulate the first three rounds of $\mathcal{P}^*\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}\right)$
  ⋮
end procedure

Going from GAME$_7$ to GAME$_8$ (see Algorithm 3.15) we perform the steps we did from GAME$_1$ to GAME$_3$ in reverse.

The next step needs some preparation. Let $\mathbf{s} := \left(1, -\sum_{k\in[N_{\text{parties}}]} r_k\right)^T$. By Definition 3.1.2 we have $\mathbf{C}_{\text{out}_j}\mathbf{s} = \mu'_j\mathbf{G}\mathbf{s} + \text{noise}_{\mathbf{s},\mu'_j}\left(\mathbf{C}_{\text{out}_j}\right)$. Taking the i^{th} row of that equation yields

$$\mathbf{C}_{\text{out}_j}[i,1] - \mathbf{C}_{\text{out}_j}[i,2] \cdot \sum_{k=1}^{N_{\text{parties}}} r_k = \mu'_j 2^{i-1} + \eta_i$$

Algorithm 3.15 GAME$_8$ for PassiveMPC

procedure GAME$_8$($1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}, y = (\mu'_j)_{j\in[N_{\text{outputs}}]}$)
  Simulate the first three rounds of $\mathcal{P}^*\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}\right)$
  $i := \lfloor \log q \rfloor$
  for $j \in [N_{\text{outputs}}]$ **do**
    $\mathsf{smdg} \xleftarrow{\$} R_{[-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}[}$
    $\mathsf{pp}_1^{(j)} := -\mu'_j 2^{i-1} + \mathsf{smdg} + \mathbf{C}_{\text{out}_j}[i,1] - \mathbf{C}_{\text{out}_j}[i,2] \cdot \sum_{k=2}^{N_{\text{parties}}} r_k$
    Modify P_1 such that in the subprotocol $\mathsf{Dec}\left(\mathsf{sk}, \mathbf{C}_{\text{out}_j}\right)$ it sends $\mathsf{pp}_1^{(j)}$ instead of pp_1.
  end for
  Simulate the forth round of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$, obtaining output z.
  return $\pi_K(z)$
end procedure

where η_i is the i^{th} component of $\mathsf{noise}_{\mathbf{s},\mu'_j}\left(\mathbf{C}_{\text{out}_j}\right)$. Stated differently this is

$$-\mu'_j 2^{i-1} + \mathbf{C}_{\text{out}_j}[i,1] - \mathbf{C}_{\text{out}_j}[i,2] \cdot \sum_{k=2}^{N_{\text{parties}}} r_k = \mathbf{C}_{\text{out}_j}[i,2] \cdot r_1 + \eta_i.$$

In the context of GAME$_8$ it follows that

$$\mathsf{pp}_1^{(j)} = \mathbf{C}_{\text{out}_j}[i,2] \cdot r_1 + \eta_i + \mathsf{smdg}.$$

Going from GAME$_8$ to GAME$_9$ we essentially replaced $\eta_i + \mathsf{smdg}$ by smdg. Lemma 3.3.3 states that $||\eta_i||_\infty \leq \left\|\mathsf{noise}_{\mathbf{s},\mu'_j}\left(\mathbf{C}_{\text{out}_j}\right)\right\|_\infty$ is bounded by some polynomial $p(B_\chi)$. As $B_\chi(\lambda)/B_{\mathsf{smdg}}(\lambda)$ is negligible, $p(B_\chi(\lambda))/B_{\mathsf{smdg}}(\lambda)$ is still negligible. Therefore we can apply Lemma 3.3.1 stating that $\eta_i + \mathsf{smdg}$ is indistinguishable from smdg.

Algorithm 3.16 GAME$_9$ for PassiveMPC

procedure GAME$_9$($1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}, y = (\mu'_j)_{j\in[N_{\text{outputs}}]}$)
  Simulate the first three rounds of $\mathcal{P}^*\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}\right)$.
  $i := \lfloor \log q \rfloor$
  for $j \in [N_{\text{outputs}}]$ **do**
    $\mathsf{smdg} \xleftarrow{\$} R_{[-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}[}$
    $\mathsf{pp}_1^{(j)} := \mathbf{C}_{\text{out}_j}[i,2]] \cdot r_1 + \mathsf{smdg}$
    Modify P^* such that in the subprotocol $\mathsf{Dec}\left(\mathsf{sk}, \mathbf{C}_{\text{out}_j}\right)$ P_1 sends $\mathsf{pp}_1^{(j)}$ instead of pp_1.
  end for
  Simulate the forth round of $\mathcal{P}^*\left(1^\lambda, (\tilde{\mathbf{x}}_k)_{k\in[N_{\text{parties}}]}\right)$, obtaining output z.
  return $\pi_K(z)$
end procedure

In GAME_9, observe that the modifications done to $\mathcal{P}^*$ don't actually change any values. All values they set are already in effect. We obtain GAME_{10} by eliminating those modifications, which is a purely syntactic change. It's easy to see that $\text{GAME}_{10}\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}\right) \equiv \pi_K\left(\mathcal{P}^*\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}\right)\right)$.

Algorithm 3.17 GAME_{10} for PassiveMPC

procedure $\text{GAME}_{10}(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]})$
  $z \leftarrow \mathcal{P}^*\left(1^\lambda, (\mathbf{x}_k)_{k\in[N_{\text{parties}}]}\right)$.
  return $\pi_K(z)$
end procedure

□

3.5 Parameter Choices

In order to base the security of our MPC protocol on the reduction from SIVP to Ring-LWE (see Theorem 2.8.1), we need to choose the parameters such that the approximate SIVP problem from the reduction is assumed to be hard. This is the case for $d \in \Omega(\lambda \log(q/B_\chi))$ [BGV11]. We use $d = \lambda\lceil\log(q/B_\chi)/2\rceil$. For fixed depth we have $\log(q/B_\chi) \in \Theta(\lambda)$, so our choice of d is the analogue of $n = \lambda^2$ used by [AJW11] for the standard LWE dimension n. We must use the error distribution χ from the reduction which is a multivariate Gaussian distribution with $B_\chi \in \omega\left(\sqrt{\log d}\right)$. Concretely we use $B_\chi \approx \log d$ rounding to the next power of two. Observe that we need to know d in order to choose B_χ, but we need to know q and B_χ in order to choose d. This is a circular definition. In practice we solve this using an iterative approach, starting with the approximation $B_\chi \approx \log(\lambda^2) = 2\log\lambda$ (rounding to the next power of two). With this we describe below how to compute the parameters B_{smdg} and q. Then we can compute d as stated in this paragraph and afterwards we check if B_χ is large enaugh. If not, we increase it and start over.

B_{smdg} must be chosen such that B_{smdg}/B_χ is negligible in the security parameter. This implies that B_{smdg} and notably q are superpolynomial in the security parameter. Without smudging q could be much smaller and the protocol thus much more efficient. While in the threshold levelled FHE scheme from [AJW11] there was smudging in three places, leading to even higher parameters, we already managed to eliminate the smudging in two of those three places by using commitments and by basing on the GSW scheme [GSW13] instead of BGV [BGV11]. We leave it as an open question whether it is possible to eliminate smudging entirely from the protocol.

The proof of the smudging lemma in [AJW11] states that the statistical difference between the two relevant distributions is bounded by B/B_{smdg} where B is in our case an upper bound for the noise level. This yields $\log(B_{\text{smdg}}/B)$ bits of security for the smudging itself. In order to achieve roughly λ bits, we can use $B_{\text{smdg}} = 2^\lambda B$. Even though the noise term $(N_{\text{parties}}+1)B_\chi^2(m+1)^{\text{dpth}}$ from Equation (3.6) is polynomial in B_χ for fixed dpth, in practice we should not ignore this polynomial factor. Therefore we first need to compute a bound B on the noise and then choose $B_{\text{smdg}} = 2^\lambda B$. Note that B depends on the depth of the circuit. For the last parameter q (Recall that $m = 2(\lfloor\log q\rfloor + 1)$ is fixed.), we need $q \equiv 1 \mod 2d$ by Theorem 2.8.1 and additionally

λ	d	q	B_χ	B_{smdg}	max dpth	pk size
128	24320	$2^{384} - 1081343$	16	2^{368}	22	≈ 1.67 GiB
256	65024	$2^{512} - 3014655$	16	2^{496}	21	≈ 7.94 GiB

Table 3.1: Example parameters for $N_{\mathsf{parties}} = 4096$ and $\lambda \in \{128, 256\}$.

Equation (3.6) must hold true where dpth is the depth of the arithmetic circuit that we wish to evaluate. Because of the large B_{smdg} relative to the noise term, Equation (3.6) essentially becomes $q > (8N_{\mathsf{parties}} + \varepsilon)B_{\mathsf{smdg}}$.

If the depth of the circuit is not predetermined, one can do it the other way around: First choose q, then choose B_{smdg} such that $q > (8N_{\mathsf{parties}} + \varepsilon)B_{\mathsf{smdg}}$, then set $B = B_{\mathsf{smdg}}/2^\lambda$ and finally compute the largest dpth such that the noise term does not exceed B; that is, $\left(N_{\mathsf{parties}} + 1\right) B_\chi^2 \cdot (m+1)^{\mathsf{dpth}} \leq B$. See Table 3.1 for two example parameter sets chosen following this policy. Note that the column showing the maximum dpth is based on our worst-case noise term. We may evaluate *every* 1-bounded arithmetic circuit of that depth, but does not mean that every circuit of higher depth must not be evaluated. For specific circuits of higher depth, one can often find a better upper bound B_{noise} for the noise level of the outputs by performing an analysis that is not merely based on the depth of the circuit. The circuit is eligible if $B_{\mathsf{noise}} \leq B_{\mathsf{smdg}}/2^\lambda$.

If the circuit to evaluate has a depth greater than allowed by these parameters (and one does not want to perform a circuit-specific analysis), then a larger q has to be chosen (which might subsequently impact the choice of d and B_χ slightly). When fixing the security parameter, then the lower bound for $\log q$ grows at most linear in dpth $\log$ dpth, i.e. we can choose $\log q \in O\left(\mathsf{dpth} \log \mathsf{dpth}\right)$. This means that q increases exponentially in dpth, which is the main problem in practice. Alternatively, when fixing dpth, then $\log q \in O\left(\lambda\right)$.

3.6 Asymptotic Complexity

In this section we analyze the asymptotic complexity of PassiveMPC dependent on the security parameter λ, on N_{parties}, on $\log q$, and on the size of the evaluated arithmetic circuit $|f|$. The size is measured as the number of gates.

We ignore the space and time complexity of the used commitment scheme $\mathcal{K}$. Note that any commitment scheme that needs space and time linear in the space of the commited value would not contribute to the asymptotic complexity anyway.

First we analyze the space complexity *per party*. Observe that the private key needs less space than a public key share and a public key share needs as much space as a ciphertext. The matrix $\mathbf{G}^{-1}(\cdot)$ used during homomorphic evaluation does not have to be computed in one piece, but rather its rows can be computed lazily. Therefore it suffices to consider the public key shares and ciphertexts in order to obtain the asymptotic space complexity. Each ciphertext (or public key share) consists of $2m \in O\left(\log q\right)$ ring elements, each of which needs space $d \log q \in O\left(\lambda \log^2 q\right)$. Therefore the space complexity of the whole protocol is in $O\left(\left(N_{\mathsf{parties}} + |f|\right) \lambda \log^3 q\right)$. For a fixed circuit this is in $O\left(N_{\mathsf{parties}} \lambda^4\right)$.

For the analysis of the time complexity (per party) we assume the worst-case which is that every gate of f is a multiplication gate. For each multiplication $\mathbf{G}^{-1}(\mathbf{C}_1)\,\mathbf{C}_2$ we have $\mathbf{G}^{-1}(\mathbf{C}_1) \in R_{[0,1]}^{m\times m}$ and $\mathbf{C}_2 \in R_q^{m\times 2}$. That amounts to $2m^2 \in O(\log^2 q)$ multiplications in R_q. We would like to do multiplications in R_q in the Chinese remainder embedding. However, $\mathbf{G}^{-1}(\mathbf{C}_1)$ can not simply be computed in the Chinese remainder embedding, because it performs a bit decomposition with respect to the coefficient embedding. Therefore, assuming the ciphertexts are present in the Chinese remainder embedding, we have to perform a change of basis into the coefficient embedding, then compute $\mathbf{G}^{-1}(\mathbf{C}_1)$ and then revert the change of basis. A change of basis of a single ring element comes at the cost of $O(d^2) = O(\lambda^2 \log^2 q) \subseteq O(\lambda \log^3 q)$ time. (Optimizations of the change of basis are possible and certainly relevant in practice, but are not relevant for the asymptotic time complexity, as we will see.) The time for changing the basis of a whole ciphertext is thus in $O(\lambda \log^4 q)$. When computing the multiplications in the Chinese remainder embedding, then each multiplication in R_q needs time in $O(d \log^2 q) = O(\lambda \log^3 q)$ when using classical multiplication or $O(\lambda \log^2 q \cdot \log\log q \cdot \log\log\log q)$ when using the Schönhage–Strassen algorithm [SS71]. Going forward we abbreviate this by $O(\lambda \log^{2+\varepsilon} q)$, so each homomorphic multiplication needs time in $O(\lambda \log^{4+\varepsilon} q)$. Here we see that the change of basis is not relevant for the asymptotic time complexity. The whole evaluation needs time in $O(|f|\lambda \log^{4+\varepsilon} q)$. The reception and addition of (the commitments of) the N_{parties} public key shares in round I and II needs time linear in their space, i.e. $O(N_{\text{parties}}\lambda \log^3 q)$, and everything not mentioned here needs less time. Altogether the asymptotic time complexity is in $O((N_{\text{parties}} + |f| \log^{1+\varepsilon} q)\lambda \log^3 q)$. For a fixed circuit this is in $O((N_{\text{parties}} + \log^{1+\varepsilon} q)\lambda \log^3 q)$.

Lastly we analyze the communication. Each party broadcasts exactly 4 messages. The public key shares need space in $O(N_{\text{parties}}\lambda \log^3 q)$, the input ciphertexts need space in $O(N_{\text{inputs}}\lambda \log^3 q)$ and the output pre-plaintexts need space in $O(N_{\text{parties}}N_{\text{outputs}}\lambda \log^2 q)$. This amounts to a total communication complexity in

$$O\big((N_{\text{parties}} + N_{\text{inputs}})\lambda \log^3 q + N_{\text{parties}}N_{\text{outputs}}\lambda \log^2 q\big). \tag{3.7}$$

Note that there is no difference between the total communication complexity and the communication complexity per party, because the communication is through a broadcast channel, so every party receives every message (except for adversarial messages which are not considered in this analysis). For a fixed circuit the communication complexity is in $O(N_{\text{parties}}\lambda^4)$.

Table 3.2 displays a summary of this section.

	Fixed circuit	Variable circuit
Space per party	$N_{\text{parties}}\lambda^4$	$(N_{\text{parties}} + \lvert f\rvert)\lambda \log^3 q$
Time	$(N_{\text{parties}} + \lambda^{1+\varepsilon})\lambda^4$	$(N_{\text{parties}} + \lvert f\rvert \log^{1+\varepsilon} q)\lambda \log^3 q$
Communication	$N_{\text{parties}}\lambda^4$	See Equation (3.7)

Table 3.2: Asymptotic complexity of PassiveMPC; $O(\cdot)$ omitted

4 From Passive to Active Security

This chapter describes the measures that can be taken in order to transform our MPC protocol into an actively secure version. Afterwards in Section 4.5 we analyze how this changes the asymptotic complexity.

In order to achieve security against active adversaries, i.e. adversaries that deviate from the protocol arbitrarily, the parties have to prove in each round that they followed the protocol correctly. We use non-interactive ZKPoKs in order to still preserve privacy. A party P_k can deviate from the protocol by choosing a malformed contribution $\mathbf{b}_k$ to the public key, by inputting a malformed ciphertext, or by broadcasting a malformed share pp_k. It turns out that choosing a malformed $\mathbf{b}_k$ does not undermine privacy because our commitment strategy ensures that the resulting "public key" is still distributed uniformly at random. Therefore we can postpone proving the correctness of $\mathbf{b}_k$ to round IV where that can be proven together with the correctness of pp_k in a single proof.

All required proofs can be reduced to proofs for the inhomogeneous short integer solution (ISIS) problem [GPV07].

Definition 4.0.1 (The ISIS problem)**.** For security parameter λ, let $n = n(\lambda), m = m(\lambda), \beta = \beta(\lambda) \in \mathbb{N}$. The search variant of the $\mathsf{ISIS}^p_{n,m,q,\beta}$ problem is as follows. Given a uniformly chosen matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times m}$ and uniformly chosen vector $\mathbf{y} \in \mathbb{Z}_q^n$, find a witness $\mathbf{x} \in \mathbb{Z}_q^m$ such that $||\mathbf{x}||_p \leq \beta$ and $\mathbf{Mx} = \mathbf{y}$. We overload notation by defining the corresponding relation

$$\mathsf{ISIS}^p_{n,m,q,\beta} := \left\{ ((\mathbf{M}, \mathbf{y}), \mathbf{x}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n \times \mathbb{Z}_q^m \;\middle|\; ||\mathbf{x}||_p \leq \beta \;\wedge\; \mathbf{Mx} = \mathbf{y} \right\}.$$

There is also a Ring variant of the ISIS problem [LPR12] which we call Ring-ISIS and where $\mathbf{M}$, $\mathbf{y}$ and $\mathbf{x}$ have entries from R_q instead of $\mathbb{Z}_q$. However, in our use cases sometimes we require that certain entries of the witness $\mathbf{x}$ are from $\mathbb{Z}_q$. To our knowledge this can not be reduced to Ring-ISIS without blowing up the matrix $\mathbf{M}$ by essentially reducing to ISIS and then interpreting the ISIS instance as a Ring-ISIS instance. Therefore we have two options. Either we reduce all required proofs to ISIS and use a proof therefor, or we use a proof for Ring-ISIS that supports our additional requirements.

Fortunately the latter exists. Our additional requirements are as follows. The secret vector $\mathbf{x}$ can have different bounds for different entries. Furthermore, some of the entries of $\mathbf{x}$ are required to be in $\mathbb{Z}_q$ rather than R_q. The proof by del Pino et al. [PLS19] is a ZKPoK for Ring-ISIS that supports these requirements, as long as all of the bounds are intervals of the form $\left[-2^{b-1}, 2^{b-1}\right[$. This is because in said proof, the coefficients of the ring elements in $\mathbf{x}$ are represented using two's complement binary representation and their range is implied by the number of bits. For this reason we require that B_{smdg} and B_χ are powers of two. See Section 4.2 for elaboration on why we chose the proof by del Pino et al. to instantiate our protocol with.

Now we show that the actual statements which the parties have to prove are of the correct form.

Firstly, a ciphertext $\mathbf{C}$ inputted by a party P_k is correct if P_k knows $t \in R_{[-B_\chi, B_\chi[}$, $\mathbf{F} \in R^{m\times 2}_{[-B_\chi, B_\chi[}$ and $\mu \in \{0,1\}$ such that $\mathbf{C} = t\mathbf{A} + \mathbf{F} + \mu\mathbf{G}$. Observe that

$$
\begin{aligned}
& \mathbf{C} = t\mathbf{A} + \mathbf{F} + \mu\mathbf{G} \\
\iff \quad & \begin{bmatrix}\mathbf{C}[\cdot,1]\\ \mathbf{C}[\cdot,2]\end{bmatrix} = \begin{bmatrix}\mathbf{A}[\cdot,1]\\ \mathbf{A}[\cdot,2]\end{bmatrix}\cdot t + \begin{bmatrix}\mathbf{F}[\cdot,1]\\ \mathbf{F}[\cdot,2]\end{bmatrix} + \begin{bmatrix}\mathbf{G}[\cdot,1]\\ \mathbf{G}[\cdot,2]\end{bmatrix}\cdot \mu \\
\iff \quad & \begin{bmatrix}\mathbf{C}[\cdot,1]\\ \mathbf{C}[\cdot,2]\end{bmatrix} = \begin{bmatrix}\mathbf{A}[\cdot,1] & I_m & & \mathbf{G}[\cdot,1]\\ \mathbf{A}[\cdot,2] & & I_m & \mathbf{G}[\cdot,2]\end{bmatrix}\begin{bmatrix} t\\ \mathbf{F}[\cdot,1]\\ \mathbf{F}[\cdot,2]\\ \mu\end{bmatrix}
\end{aligned}
\tag{4.1}
$$

Hereby we have varying bounds for the coefficients of the ring elements in the secret vector, but the bounds are powers of two as mentioned above.

Secondly, if a party P_k broadcasts its share pp_k, then P_k has to prove that $\mathsf{pp}_k = \mathbf{C}[i,2]\cdot r_k + \mathsf{smdg}_k$ was computed correctly and also that the secret r_k therein is the same secret as in $\mathbf{b}_k = \mathbf{a}\cdot r_k + \mathbf{e}_k$. Observe that

$$
\mathsf{pp}_k = \mathbf{C}[i,2]\cdot r_k + \mathsf{smdg}_k = \begin{bmatrix}\mathbf{C}[i,2] & 1\end{bmatrix}\begin{pmatrix} r_k \\ \mathsf{smdg}_k\end{pmatrix}
$$

and that

$$
\mathbf{b}_k = \mathbf{a}\cdot r_k + \mathbf{e}_k = \begin{bmatrix}\mathbf{a} & I_m\end{bmatrix}\begin{bmatrix} r_k \\ \mathbf{e}_k\end{bmatrix}.
$$

Hence it suffices for P_k to prove that

$$
\begin{bmatrix}\mathsf{pp}_k\\ \mathbf{b}_k\end{bmatrix} = \begin{bmatrix}\mathbf{C}[i,2] & 1 & \\ \mathbf{a} & & I_m\end{bmatrix}\begin{bmatrix} r_k\\ \mathsf{smdg}_k\\ \mathbf{e}_k\end{bmatrix}.
\tag{4.2}
$$

Again we have varying bounds for the coefficients in the secret vector $\begin{bmatrix} r_k & \mathsf{smdg}_k & \mathbf{e}_k^T\end{bmatrix}^T$; that is, smdg_k must have coefficients in $\left[-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}\right[$ and r_k as well as entries of $\mathbf{e}_k$ must have coefficients in $\left[-B_\chi, B_\chi\right[$.

4.1 Reductions to Canonical Ring-ISIS

In Chapter 4 we argued that a party has to prove that they know a vector $\mathbf{x} \in R_q^m$ that satisfies some linear equation where the coefficients of the ring elements in $\mathbf{x}$ are bounded by (potentially different) intervals of the form $\left[-2^{b-1}, 2^{b-1}\right[$. Additionally, some of the entries of $\mathbf{x}$ are required to be in $\mathbb{Z}_q$ rather than R_q. We will now reduce these instances to instances in the canonical form of ISIS as defined in Definition 4.0.1. This is just in case anybody wants to instantiate our protocol with another ZKPoK for canonical ISIS, should for example a more efficient ZKPoK for ISIS be discovered in the future. The reduction is not relevant for our instantiation of the protocol using the proof by del Pino et al. [PLS19].

We perform the reduction in three steps. After the first reduction, all entries of $\mathbf{x}$ are over $\mathbb{Z}_q$ instead of R_q and the bounds are preserved. The second reduction does the same to $\mathbf{M}$ and $\mathbf{y}$. After the third reduction, $\mathbf{x}$ has only entries from $\{-1, 0, 1\}$. This yields an instance of ISIS^∞ with $\beta = 1$. In the first two reductions we imply the bounds for the sake of brevity; it's easy to see that they are preserved.

Lemma 4.1.1. *Let* $\mathbf{M}_1 \in \mathbb{Z}_q^{n \times m_1}$, $\mathbf{M}_2 \in R_q^{n \times m_2}$, $\mathbf{y} \in R_q^n$ *and define*

$$\mathbf{M}' := \begin{bmatrix} \mathbf{M}_1 & \left(\mathbf{M}_2 \otimes \left(X^0, \ldots, X^{d-1}\right)\right) \end{bmatrix} \in R_q^{n \times (m_1 + m_2 d)}.$$

Then

$$\exists (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{Z}_q^{m_1} \times R_q^{m_2} : \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \mathbf{y} \quad \iff \quad \exists \mathbf{x}' \in \mathbb{Z}_q^{m_1 + m_2 d} : \mathbf{M}'\mathbf{x}' = \mathbf{y}$$

and given $(\mathbf{x}_1, \mathbf{x}_2)$ *or* $\mathbf{x}'$ *one can efficiently compute such* $\mathbf{x}'$ *or* $(\mathbf{x}_1, \mathbf{x}_2)$*, respectively.*

Proof. Given $\mathbf{x}_1$ and $\mathbf{x}_2$, write $\mathbf{x}_2 =: \left(z_1, \ldots, z_{m_2}\right)^T$ and define

$$\mathbf{x}' := \begin{bmatrix} \mathbf{x}_1 & \mathsf{coefficents}(z_1) & \ldots & \mathsf{coefficents}\left(z_{m_2}\right) \end{bmatrix}^T \in \mathbb{Z}^{m_1 + m_2 d}.$$

For the other direction do the same in reverse. □

Lemma 4.1.2. *Let* $\mathbf{M} \in R_q^{n \times m}$ *and* $\mathbf{y} \in R_q^n$*. Decompose* $\mathbf{M} := \sum_{i=0}^{d-1} \mathbf{M}_i X^i$ *with* $\mathbf{M}_i \in \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{y} := \sum_{i=0}^{d-1} \mathbf{y}_i X^i$ *with* $\mathbf{y}_i \in \mathbb{Z}_q^n$*. Define*

$$\mathbf{M}' := \begin{bmatrix} \mathbf{M}_0 \\ \vdots \\ \mathbf{M}_{d-1} \end{bmatrix}, \quad \mathbf{y}' := \begin{bmatrix} \mathbf{y}_0 \\ \vdots \\ \mathbf{y}_{d-1} \end{bmatrix}.$$

Then for all $\mathbf{x} \in \mathbb{Z}_q^m$ *it holds true that*

$$\mathbf{M}\mathbf{x} = \mathbf{y} \quad \iff \quad \mathbf{M}'\mathbf{x} = \mathbf{y}'.$$

Proof. Observe that

$$\begin{aligned} \mathbf{M}\mathbf{x} = \mathbf{y} \quad &\iff \quad \sum_{i=0}^{d-1} \mathbf{M}_i \mathbf{x} X^i = \sum_{i=0}^{d-1} \mathbf{y}_i X^i \\ &\iff \quad \forall i \in \{0, \ldots, d-1\} : \mathbf{M}_i \mathbf{x} = \mathbf{y}_i \\ &\iff \quad \mathbf{M}'\mathbf{x} = \mathbf{y}'. \end{aligned}$$

Thereby the second equivalence is obtained by comparison of coefficients. □

Lemma 4.1.3. *Let* $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{y} \in \mathbb{Z}_q^n$*. For all* $j \in [m]$ *let lengths* $b_j \in \mathbb{N}$ *be given. Define* $m' := \sum_{j=0}^{m} b_j$ *and*

$$\begin{aligned}\mathbf{M}' &:= \left[\mathbf{M}[i,j] \cdot \left(2^0, \ldots, 2^{b_j-1}\right)\right]_{i\in[n], j\in[m]} \\ &= \left[\left(\mathbf{M}[^*,1] \otimes \left(2^0, \ldots, 2^{b_1-1}\right)\right) \quad \cdots \quad \left(\mathbf{M}[^*,m] \otimes \left(2^0, \ldots, 2^{b_m-1}\right)\right)\right] \\ &\in \mathbb{Z}_q^{n \times m'}.\end{aligned}$$

Then

$$\begin{aligned}&\exists \mathbf{x} = (z_1, \ldots, z_m)^T \in \mathbb{Z}_q^m : \left(\mathbf{M}\mathbf{x} = \mathbf{y} \wedge \forall j : z_j \in \left]-2^{b_j}, 2^{b_j}\right[\right) \\ \iff \quad &\exists \mathbf{x}' \in \{-1,0,1\}^{m'} : \mathbf{M}'\mathbf{x}' = \mathbf{y}\end{aligned}$$

and given $\mathbf{x}$ *or* $\mathbf{x}'$ *one can efficiently compute such* $\mathbf{x}'$ *or* $\mathbf{x}$*, respectively.*

Proof. Given $\mathbf{x}$, for all $j \in [m]$, perform a bit decomposition $z_j =: 2^0 z_{j,0} + \cdots + 2^{b_j-1} z_{j,b_j-1}$ with $z_{j,e} \in \{-1,0,1\}$. Define $\mathbf{x}' := \left[\left(z_{j,0}, \ldots, z_{j,b_j-1}\right)\right]_{j\in[m]}$. For the other direction do the same in reverse. □

4.2 Discrete-Log-Based Zero-Knowledge Proof of Knowledge

The ZKPoK by del Pino et al. [PLS19], which we subsequently call the PLS proof, fits our requirements stated in this chapter. There exist other proofs that also support these requirements [BCK+14] or can be used after our reduction from Section 4.1 [LNSW12]. We will now explain why we chose the PLS proof over those others.

In Section 4.1 we reduced the proofs required for our protocol to proofs for the ISIS problem. However, the first two steps of the reduction each blow up the matrix by a factor of d. Even after considering that the resulting matrix is no longer over R_q but rather over $\mathbb{Z}_q$, this is still in total a blowup by a factor of d. Such a blowup is undesirable because the complexity of a proof typically depends on the size of the matrix [BCK+14; LNSW12; PLS19]. The PLS proof circumvents the blowup by evaluating the equation $\mathbf{M}\mathbf{x} = \mathbf{y}$ at a random point $\alpha \xleftarrow{\$} \mathbb{Z}_q$ and proving the resulting equation

$$\mathbf{M}(\alpha)\mathbf{x} = \mathbf{y}(\alpha) \tag{4.3}$$

instead. This essentially replaces the second step of our reduction from Section 4.1 as in Equation (4.3) $\mathbf{M}(\alpha)$ and $\mathbf{y}(\alpha)$ are over $\mathbb{Z}_q$. Observe that in Equation (4.3) the blowup does not occur. Soundness is preserved by the Schwartz-Zippel lemma [Sch80; Zip79] which states that if Equation (4.3) holds true for a random $\alpha \xleftarrow{\$} \mathbb{Z}_q$, then $\mathbf{M}\mathbf{x} = \mathbf{y}$ holds true with probability $\geq 1 - 2(d-1)/q$. In the PLS proof this circumvention of the blowup leads to a significantly reduced time complexity for creating and verifying proofs.

The main reason why we chose the PLS proof is its small proof size. Firstly, one PLS proof has size only logarithmic in the size of $\mathbf{x}$ and $\mathbf{y}$; i.e. the proof size is in $\Theta\big(\log(|G|)\log((m+n)\,d\log q)\big)$ for $\mathbf{x} \in R_q^m$ and $\mathbf{y} \in R_q^n$. Hereby $|G|$ is a cyclic group where the discrete-log assumption holds true. Secondly, the knowledge error is rather small at $1/d$ and hence the proof only needs to be repeated in the magnitude of a dozen times [PLS19].

The PLS proof is statistical zero-knowledge. Its validity however is based on the discrete-log assumption in some large enaugh cyclic group G. Thereby $|G|$ can be much larger than q, so the usage of PLS proofs does not force us to use a larger q than in the passively secure MPC protocol.

4.3 Non-Interactive Zero-Knowledge

Formally, the IPS constructed by del Pino et al. [PLS19] is a PoK but not yet perfect zero-knowledge. This is expected because it only has the *special honest verifier zero-knowledge* property which intuitively means that an honest verifier learns nothing about the secret. A dishonest verifier on the other hand could undermine privacy by drawing its challenges from some distribution violating the PLS protocol. There are standard approaches to transform a *special honest verifier zero-knowledge* PoK into a ZKPoK. One such approach is the Fiat-Shamir transform [FS86] which yields a non-interactive ZKPoK in the ROM [PS96]. In practice the random oracle is replaced by a hash function and therefore the Fiat-Shamir transform is also called Fiat-Shamir heuristic.

The prover in the Fiat-Shamir transform of a ZKPoK the prover simulates an execution of that ZKPoK and then sends a single message consisting of the transcript. Each time it's the verifiers turn in the simulation, the random oracle is accessed with the transcript so far and the obtained randomness is used as random coins for the continuation of the simulated verifier. Given such a non-interactive ZKPoK, anybody can verify it by rerunning the simulation taking the prover's messages from the transcript. The verification is successful if the reran transcript matches the original and the verifier in the simulation accepted the proof.

The random coins used by the verifier to compute a challenge are always immediately available to the prover in the Fiat-Shamir transform. In order to preserve validity it's important that the original protocol already had the property that the coins of the verifier are immediately disclosed. This is the case in the PLS proof. Therein for each challenge the verifier draws the challenge uniformly at random and sends it to the prover.

4.4 Our Actively Secure MPC Protocol

In this section we define the actively secure version of our MPC protocol.

When working with a non-interactive IPS $\mathcal{I} = (Q, V)$, we don't formalize Q, V as ITMs. Instead we expect Q to output what would otherwise be its only message and we give that output as an auxiliary input to V. In our case $p \leftarrow Q((\mathbf{M}, \mathbf{y}), \mathbf{x})$ outputs a "proof" p certifying that $\mathbf{Mx} = \mathbf{y}$ (where $\mathbf{x}$ is a short secret) and $b \leftarrow V((\mathbf{M}, \mathbf{y}), p)$ outputs a bit $b \in \{0, 1\}$.

Definition 4.4.1 (Our actively secure MPC protocol)**.** Let λ be the security parameter. Let $\mathcal{I} = (Q, V)$ be a non-interactive IPS for Ring-ISIS^∞ or an appropriate variation thereof as described in the beginning of this chapter. Let $\mathcal{K} = (\mathsf{Gen}_\mathcal{K}, \mathsf{Com}_\mathcal{K})$ be a commitment scheme. Let f be an 1-bounded arithmetic circuit over R_q. Let N_{inputs} be the number of input gates and dpth be the depth of f. For λ, dpth as specified above, let $d, R, q, \chi, B_{\mathsf{smdg}}$ be as in Definition 3.3.1. Furthermore, there are N_{parties} parties $P_1, \ldots, P_{N_{\mathsf{parties}}}$ and $\phi : [N_{\mathsf{inputs}}] \to [N_{\mathsf{parties}}]$ assigns each input gate

to a party. For each party P_k, the protocol input has the form $\mathbf{x}_k := (\mu_i)_{i \in \phi^{-1}(k)}$. In the ROM, our activley secure MPC protocol $\mathsf{ActiveMPC}_{\mathcal{I},\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},f,\phi}\left(1^\lambda, (\mathbf{x}_k)_{k \in [N_{\mathsf{parties}}]}\right)$ is defined as follows, utilizing $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}) = \mathsf{TFHE}_{\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},N_{\mathsf{parties}}}$ from Definition 3.3.1.

Round I. The parties run the first round of the key generation protocol $\mathsf{Gen}\left(1^\lambda, \mathsf{dpth}\right)$. (This is the same as in the passively secure version.)

Round II. The parties run the second round of the key generation protocol, such that every party obtains the public key pk and their share of the secret key sk. (This is also the same as in the passively secure version.)

Round III. For each $i \in [N_{\mathsf{inputs}}]$: Let in_i be the i^{th} input gate. Party $P_{\phi(i)}$ computes $\mathbf{C}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_i)$, utilizes Q to compute a proof p_i certifying that Equation (4.1) holds true (for $\mathbf{C} = \mathbf{C}_i$) and broadcasts $(i, \mathbf{C}_i, p_i)$. Each party then verifies the proofs; that is: If there exists $j \in \left[N_{\mathsf{inputs}}\right]$ such that V on input the corresponding matrix, vector and p_j outputs 0, then abort.

Round IV. The parties compute $\left(\mathbf{C}_{\mathsf{out}_1}, \ldots, \mathbf{C}_{\mathsf{out}_{N_{\mathsf{outputs}}}}\right) \leftarrow \mathsf{Eval}\left(\perp, f, \left(\mathbf{C}_{\mathsf{in}_1}, \ldots, \mathbf{C}_{\mathsf{in}_{N_{\mathsf{inputs}}}}\right)\right)$. In parallel for each $i \in \left[N_{\mathsf{outputs}}\right]$ they perform the one-round decryption protocol $\mu_i' \leftarrow \mathsf{Dec}\left(\mathsf{sk}, \mathbf{C}_{\mathsf{out}_i}\right)$ and for each decryption they utilize Q to compute and broadcast a proof p_i' certifying that Equation (4.2) holds true for their decryption share. Note that all of the messages of this round are sent at once, so this is indeed only a single round. Each party then verifies the proofs p_j', $j \in \left[N_{\mathsf{outputs}}\right]$, analogously to round III. Parties which did not abort return $\left(\mu_1', \ldots, \mu_{N_{\mathsf{outputs}}}'\right)$.

We give the following two theorems without proof. They follow intuitively because the ZKPoKs prevent active adversaries from deviating from the protocol (or otherwise the honest parties abort), so the security is the same as in the presence of only passive adversaries.

Theorem 4.4.1. *Let $B_\chi \in \mathbb{N}$ such that χ is B_χ-bounded. If $\mathcal{I}$ is a PoK with negligible knowledge error and for all $\lambda \in \mathbb{N}$ Equation (3.6) holds true, then $\mathsf{ActiveMPC}_{\mathcal{I},\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},f,\phi}$ provides correctness with respect to f (as defined in Definition 2.6.7).*

Theorem 4.4.2. *If*

- *$\mathcal{I}$ is a ZKPoK with negligible knowledge error,*
- *$\mathcal{K}$ is perfectly hiding, computationally binding and non-malleable,*
- *the $\mathsf{RLWE}_{d,q,\chi}$ assumption holds true and*
- *B_{smdg}/B_χ is negligible,*

then $\mathsf{ActiveMPC}_{\mathcal{I},\mathcal{K},d,q,\chi,B_{\mathsf{smdg}},f,\phi}$ provides computational privacy (as defined in Definition 2.6.8).

4.5 Asymptotic Complexity of the Proofs

We will now analyze how the asymptotic complexity of ActiveMPC when instantiated with the PLS proof differs from the asymptotic complexity of PassiveMPC.

An assessment of the PLS proof [PLS19] yields that the prover and the verifier in a proof of $\mathbf{Mx} = \mathbf{y}$ with $\mathbf{M} \in R_q^{h \times w}$ compute at most $O(b + hd \log q)$ exponentiations where b is the number of bits used to encode $\mathbf{x}$. Hereby each coefficient can have a different number of bits, depending on its (power of two) bound. (This way it is ensured that a coefficient does not exceed its bound.) As described in Section 3.5, we have $d \in O(\lambda \log q)$ and $\lambda \in O(\log q)$, so the number of exponentiations is simplified to $O(b + h\lambda \log^2 q)$. The PLS proofs in round III have $h = 2m$ and $w = 2m + 2$. The PLS proofs in round IV have $h = m + 1$ and $w = m + 2$. In either case we have $h, w \in O(\log q)$. $\mathbf{x}$ consists of w ring elements, so we have $b \leq wd \log q \in O(\lambda \log^3 q)$. Consequently the number of exponentiations (per proof or verification) can be further simplified to $O(\lambda \log^3 q)$. Each party creates $1 + N_{\text{outputs}}$ of such PLS proofs and verifies the $(N_{\text{parties}} - 1)(1 + N_{\text{outputs}})$ PLS proofs of the other parties. That amounts to a total of $O(N_{\text{parties}} N_{\text{outputs}} \lambda \log^3 q)$ exponentiations.

Those exponentiations take a group element to the power of some element from $\{0, \ldots, q - 1\}$. Using the fast exponentiation algorithm, each exponentiation takes $O(\log q)$ group operations. The time needed for each group operation depends on the group. One must use a group where the discrete-log assumption holds true [PLS19].

Let $g(q)$ be the time needed for a group operation in the group used for the PLS proof. Then the time complexity of the creation and verification of PLS proofs in the ActiveMPC protocol is

$$O(N_{\text{parties}} N_{\text{outputs}} g(q) \lambda \log^4 q). \tag{4.4}$$

Clearly it holds true that $g(q) \in \Omega(\log q)$, because the algorithm computing the group operation has to read the whole encoding of the operands and the group must be at least of cardinality q. Comparing Equation (4.4) to Table 3.2, we follow that for a fixed circuit the time complexity for the PLS proofs is simultaneously the time complexity for the whole protocol.

The space and communication complexity of the ActiveMPC protocol is the same as of the PassiveMPC protocol, because the PLS proofs are quite small. Table 4.1 displays the space, time and communication complexity of the ActiveMPC protocol.

	Fixed circuit	Variable circuit
Space per party	$N_{\text{parties}} \lambda^4$	$(N_{\text{parties}} + \lvert f \rvert) \lambda \log^3 q$
Time	$N_{\text{parties}} N_{\text{outputs}} g(q) \lambda^5$	$(N_{\text{parties}} N_{\text{outputs}} g(q) + \lvert f \rvert \log^{\varepsilon} q) \lambda \log^4 q$
Communication	$N_{\text{parties}} \lambda^4$	See Equation (3.7)

Table 4.1: Asymptotic complexity of ActiveMPC; $O(\cdot)$ omitted

5 Verifiable Tally-Hiding E-Voting

In this chapter we introduce E-voting in general as well as the two concepts of *verifiable* and *tally-hiding* E-voting. In Section 5.3 we then present our E-voting protocol based on ActiveMPC.

An E-voting system is essentially an MPC protocol that typically proceeds as follows. There is a subset of the parties called *voters*. Each voter computes and then broadcasts a ballot which contains the voter's choice (in encrypted form). Another disjoint subset of the parties, called the *trustees* or *voting authorities*, then tally the ballots. In the formalization from [Mül19] the voter might use some voter supporting device (VSD) in order to compute his ballot. We don't consider VSDs because we simply consider a voter and its VSD to be a single ppt ITM. In modern E-voting systems the tallying might be done by shuffling the ballots using mix-nets and only then decrypting them. This way it stays secret which choice belongs to which voter. However, when using this approach it will typically be public which candidate got how many votes. Another possibility is to combine the encrypted ballots using homomorphic encryption and to decrypt the result. The homomorphic evaluation corresponds to the evaluation of a function (called *result function*) on the (plaintext) choices where only the output of that function is made public. Many different result functions are possible and sensible. Our E-voting system follows this homomorphic encryption approach. After tallying, the trustees broadcast the tallying result.

Following [Mül19], we say that there are N_{voters} voters denoted by $V_1, \ldots, V_{N_{\mathsf{voters}}}$ and N_{trustees} trustees denoted by $T_1, \ldots, T_{N_{\mathsf{trustees}}}$. There is a set Ch of possible choices and each voter V_k takes her choice $\mathsf{ch}_k \in \mathsf{Ch}$ as input. At the end of the protocol execution all parties output a result res which belongs to the so-called *result space* Res. The flow of the protocol can vary between different E-voting systems. As depicted above, typically the voters run some voting procedure, broadcast their ballots, and afterwards the trustees run some tallying procedure.

5.1 Verifiability

Many E-voting systems have vulnerabilities that make it possible for an adversary to manipulate the election result [Mül19]. In this section we define the modern security property called *end-to-end verifiability*. See [Mül19] for a more in-depth description of this concept as we go over the definition relatively quickly. Intuitively an E-voting system is end-to-end verifiable if an arbitrary (possibly external) party can check whether the result of the tallying is correct or at least not too incorrect. We formalize this by adding a judge J to the protocol which runs a judging procedure and has local output from $\mathsf{Res} \cup \{\mathsf{reject}\}$. J is neither a voter nor a trustee. Her task is to judge whether or not the current protocol run achieves some goal γ. Thereby γ is formalized as a set of allowed protocol transcripts. She must do so provided only public information; that is, she sends no messages and can only receive messages from the broadcast channel. The output reject means that the judge rejects the run. We denote by "$J : \mathsf{reject}$" or "$J : \mathsf{accept}$" the events that J outputs $= \mathsf{reject}$ or $\neq \mathsf{reject}$, respectively.

Over the probability space of a protocol $\mathcal{A}(x)$ we use the random variable transcript that maps to the transcript of the protocol run. Our verifiability definition is from [KTV10], simplified to match our computational model such that we do not have to introduce their KTV framework.

Definition 5.1.1 (Verifiability)**.** Let $\mathcal{P} = \left(P_1, \ldots, P_{N_{\text{parties}}}\right)$ be an MPC protocol and let $J = P_j$ be one of its parties which can not send messages and can only receive messages from the broadcast channel. Let γ be a goal (i.e. a set of protocol transcripts) and let $\delta \in [0,1]$. We define that $\mathcal{P}$ is (γ, δ)*-verifiable by the judge* J iff. for all $K \subseteq \left[N_{\text{parties}}\right]$ with $j \notin K$ and for all tuples of ppt adversaries $\left(P_k^*\right)_{k \in K}$, there exists $n_0 \in \mathbb{N}$ such that for all input tuples x and $\lambda \geq n_0$ it holds true that

$$\mathsf{Pr}[\mathsf{transcript} \notin \gamma \wedge J : \mathsf{accept}] \leq \delta$$

over the probability space of $\mathcal{P}^*\left(1^\lambda, x\right)$. Hereby, $\mathcal{P}^*$ is the MPC protocol derived from $\mathcal{P}$ by replacing P_k by P_k^* for each $k \in K$.

Note that any MPC protocol with a judge that always rejected would suffice for this definition. Therefore one usually wants a completeness property in addition to verifiability.

Using the general definition of verifiablity one can define many different kinds of verifiability. One kind is end-to-end verifiability of which several different definitions in different computational models exist [Mül19]. We use the definition from [Mül19] and adapt it to our notation. End-to-end verifiability is parameterized by a boolean formula $\varphi : 2^{[N_{\text{parties}}]} \to \{0,1\}$ over the parties that defines which combinations of parties can be dishonest. The trust assumption is that $\varphi(K) = 1$ for the set $K \subseteq \left[N_{\text{parties}}\right]$ of dishonest parties' indices.

Definition 5.1.2 (End-to-end verifiability)**.** Let $\mathcal{P} = \left(P_1, \ldots, P_{N_{\text{parties}}}\right)$ be an MPC protocol and let $f : \left(\{0,1\}^*\right)^{N_{\text{parties}}} \to \{0,1\}^*$ be a function. Let $\varphi : 2^{[N_{\text{parties}}]} \to \{0,1\}$ be a boolean formula over the parties. For any transcript t, let

$$\Gamma(t) := \left\{ K \subseteq \left[N_{\text{parties}}\right] \;\middle|\; \exists \text{ tuple of ppt adversaries } \left(P_k^*\right)_{k \in K} \right.$$
$$\left. \text{and } \mathsf{Pr}[\mathsf{transcript} = t] > 0 \text{ over the probability space of } \mathcal{P}^*\left(1^\lambda, x\right) \right\}$$

where x is the input tuple (which is encoded in t). Observe that $\Gamma(t)$ is closed under intersection. Let $K(t)$ be the minimal element of $\Gamma(t)$. (Intuitively $K(t)$ contains the indices of parties that actually *behaved* dishonestly.) We construct the transcript set γ as follows. A transcript t is in γ iff. one of the following holds true.

- $\varphi(K(t)) = 0$.
- Or there exists an input tuple x' such that $\pi_{\left[N_{\text{parties}}\right] \setminus K(t)}(x) = \pi_{\left[N_{\text{parties}}\right] \setminus K(t)}(x')$ (That is, replacing x by x' does not change any honest party's choice.) and in t all honest parties P_k, $k \notin K(t)$, have output $f(x')$.

For any $\delta \in [0,1]$, we define that $\mathcal{P}$ is (φ, δ)*-end-to-end verifiable with respect to* f iff. $\mathcal{P}$ is (γ, δ)-verifiable.

Observe that we do not demand that the honest parties actually output the correct result. Instead, the judge can check whether the honest parties did output the correct result. Consider a protocol run of a $(\varphi, 0)$-end-to-end verifiable protocol. If the judge accepts it follows (for sufficiently large security parameter) that $\mathsf{transcript} \in \gamma$. Since $\mathsf{transcript}$ is the transcript of the considered protocol run, we have $K \in \Gamma(\mathsf{transcript})$ for the set K of dishonest parties' indices. This means that either the trust assumption is violated ($\phi(K) = 0$) or the honest parties have correct output. In short: For $\delta = 0$ and sufficiently large security parameter, if the trust assumption is not violated and the judge accepts, then the honest parties have correct output. But the judge might also reject meaning that the honest parties might or might not not have correct output. In practice one can often not achieve $\delta = 0$. For small $\delta > 0$ there is a small probability for deviation from the above statement.

The following theorem states that any correct MPC protocol is can trivially be made end-to-end verifiable for the trust assumption that there is an honest majority.

Theorem 5.1.1. *Let $\mathcal{P} = \left(P_1, \ldots, P_{N_{\mathsf{parties}}}\right)$ be an MPC protocol that is correct (as defined in Definition 2.6.7) with respect to some function f. For $i \in \left[N_{\mathsf{parties}}\right]$, let P'_i be a party that runs P_i and at the end broadcasts her output. Let J be a judge that outputs the y that is the output of more than other $N_{\mathsf{parties}}/2$ parties or $\perp$ if no such y exists. Set*

$$\begin{aligned} \varphi : 2^{\left\lceil N_{\mathsf{parties}} \right\rceil + 1} &\to \{0, 1\} \\ K &\mapsto 1_{|K| < N_{\mathsf{parties}}/2}. \end{aligned}$$

Then $\left(P'_1, \ldots, P'_{N_{\mathsf{parties}}}, J\right)$ is $(\varphi, 0)$-end-to-end verifiable with respect to f.

This already implies that we can trivially make our actively secure MPC protocol end-to-end verifiable for an honest majority. However, we will not prove Theorem 5.1.1, because we will instead use the following more significant result.

Theorem 5.1.2. *Let $\mathcal{P} = \mathsf{ActiveMPC}_{\mathcal{I}, \mathcal{K}, d, q, \chi, B_{\mathsf{smdg}}, f, \phi}$ for any parameter set allowed by Definition 4.4.1 and let N_{parties} be the number of parties of $\mathcal{P}$. If $\mathcal{P}$ provides correctness with respect to f and $\mathcal{I}$ is a PoK with negligible knowledge error κ, then there exists a judge J such that for all $\delta > 0$ and tautology $\varphi(\cdot) = 1$, $\mathcal{P}' := \left(P_1, \ldots, P_{N_{\mathsf{parties}}}, J\right)$ is (φ, δ)-end-to-end verifiable with respect to f. Furthermore, if there is besides J at least one other honest party that does not abort, then J does not reject.*

Proof idea. J waits for the other parties to finish their computation. If at least one other party aborts, then J returns reject. (Notice that with the abort of any party, all honest parties except for J also abort.) Otherwise J proceeds as follows. She computes the public key from the messages that have been broadcast in round II. Then she verifies the non-interactive PoKs for all of the input ciphertexts broadcast in round III. From those ciphertexts she computes the output ciphertexts. Afterwards she verifies the non-interactive PoKs broadcast in round IV. If any verification failed, then she returns reject. Otherwise she computes the protocol output from the decryption shares broadcast in round IV and returns the output. □

The tautology $\varphi(\cdot) = 1$ means that there is no trust assumption at all. In other words, E-voting systems based on ActiveMPC are, under the assumption stated in Theorem 5.1.2, end-to-end verifiable even if there is not a single honest party besides the judge. And the judge performs her computation only based on publicly available information, as she neither sends any messages nor interacts directly with any other party, so anybody could compute the judging procedure.

5.2 Tally-Hiding E-Voting

In an E-voting system where the choices are simply mixed and then decrypted it becomes public how often which candidate was chosen. The same is the case for a system where same choices are added up, for example using homomorphic encryption, and then decrypted. This might be undesired, especially if N_{voters} is small. In small (e.g. boardroom) votes it happens regularly that one candidate gets zero votes. If this information is made public, it reveals that every single voter did not vote for that candidate. Often times it is sufficient to know which candidate got the most votes or what the order of candidates sorted by the number of votes is. An E-voting system that has a result function, which does not reveal the tallies, is called *tally-hiding*. Such result functions can obviously be computed using our MPC protocol. In the next section we describe how to employ this in order to construct an end-to-end verifiable and tally-hiding E-voting system.

5.3 Our Verifiable Tally-Hiding E-Voting System

In this section we finally define our E-voting system. Subsequently we argue how it can be made end-to-end verifiable.

Definition 5.3.1. Let $f : \mathsf{Ch}^{N_{\text{voters}}} \to \mathsf{Res}$ be a result function given as 1-bounded arithmetic circuit and let ActiveMPC be some instantiation of our actively secure MPC protocol from Definition 4.4.1 with circuit f. Then in our E-voting system $\mathsf{EV}_f = \left(V_1, \ldots, V_{N_{\text{voters}}}, T_1, \ldots, T_{N_{\text{trustees}}}\right)$ a run $\mathsf{EV}\left(1^\lambda, x\right)$ proceeds as follows.

The parties $T_1, \ldots, T_{N_{\text{trustees}}}$ run $\mathsf{ActiveMPC}\left(1^\lambda\right)$ with the following change. The inputs and non-interactive PoKs in round III are not provided by the trustees, but instead by the voters. We have $N_{\text{inputs}} = N_{\text{voters}}$ and each voter V_i, $i \in [N_{\text{voters}}]$ provides the input $(i, \mathbf{C}_i, p_i)$ where $\mathbf{C}_i$ is the ciphertext of ch_i and p_i is the PoK. The voters compute the protocol output from the decryption shares broadcast in round IV and return the output. (This corresponds to the output computation of the judging procedure defined in the proof idea to Theorem 5.1.2.)

For brevity, in Definition 4.4.1 we did not distinguish between trustees (i.e. parties that participate in key generation and decryption) and parties that provide inputs. When making this distinction, the computational privacy analysis works analogously for the case where at least one trustee is honest. If no trustee is honest, then there is no computational privacy. Furthermore, in the correctness analysis we now have to consider whether the voters (and not only the trustees) have correct output. It turns out that this is not guaranteed at all, because any dishonest trustee can deliver a malformed decryption share in round IV. Therefore EV_f is not a correct (as defined in Definition 2.6.7) MPC protocol. However, if the voters were to run the full judging procedure (instead of only the output

computation), defined in the proof idea to Theorem 5.1.2, and abort if the judging procedure rejects, then EV_f would — under the assumptions stated in Theorem 4.4.1 — provide correctness even if an arbitrary number of trustees and/or voters are dishonest.

Note that the N_{parties} parameter used in the analysis of ActiveMPC is always the number of trustees. It does not matter how many different parties provide input.

Theorem 5.3.1. *Let* $\mathsf{EV}_f = \left(V_1, \ldots, V_{N_{\mathsf{voters}}}, T_1, \ldots, T_{N_{\mathsf{trustees}}}\right)$ *as defined in Definition 5.3.1 and let* $\mathsf{EV}'_f := \left(V_1, \ldots, V_{N_{\mathsf{voters}}}, T_1, \ldots, T_{N_{\mathsf{trustees}}}, J\right)$ *using the judge J from the proof idea to Theorem 5.1.2. If for the used instantiation of* ActiveMPC *the assumptions stated in Theorem 4.4.1 hold true, then* EV'_f *is (φ, δ)-end-to-end verifiable with respect to f for all $\delta > 0$ and tautology $\varphi(\cdot) = 1$.*

Proof idea. Let $\delta > 0$, $\varphi(\cdot) = 1$ and let $\tilde{\mathsf{EV}}_f$ be the variation of EV_f where the voters run the full judging procedure and abort if the judging procedure rejects. $\tilde{\mathsf{EV}}_f$ provides correctness, so the protocol $\tilde{\mathsf{EV}}'_f = \left(\tilde{V}_1, \ldots, \tilde{V}_{N_{\mathsf{voters}}}, T_1, \ldots, T_{N_{\mathsf{trustees}}}, J\right)$ obtained from $\tilde{\mathsf{EV}}_f$ by adding the judge J is $(\varphi, \delta/2)$-end-to-end verifiable with respect to f. Observe that we obtain EV'_f from $\tilde{\mathsf{EV}}'_f$ by reverting the replacements of the voters; that is, we replace the voters by the original voters which do not run the full judging procedure and instead perform only the output computation. We now prove that this EV'_f is (φ, δ)-end-to-end verifiable with respect to f.

Let $\gamma, \tilde{\gamma}$ be the corresponding transcript sets from Definition 5.1.2 for $\mathsf{EV}'_f, \tilde{\mathsf{EV}}'_f$, respectively. We take $K(\cdot), \tilde{K}(\cdot)$ from Definition 5.1.2 likewise. $\tilde{\mathsf{EV}}'_f$ is $(\gamma, \delta/2)$-verifiable and we have to prove that EV'_f is $(\tilde{\gamma}, \delta)$-verifiable. Let $K' \subseteq [N_{\mathsf{voters}} + N_{\mathsf{trustees}}]$ and let $\left(P^*_k\right)_{k \in K'}$ be a tuple of ppt adversaries. Let $\mathcal{P}^*$ (resp. $\tilde{\mathcal{P}}^*$) be the protocol obtained from EV'_f (resp. $\tilde{\mathsf{EV}}'_f$) by replacing V_k (resp. $\tilde{V}_k$) by P_k for each $k \in [N_{\mathsf{voters}}]$ and T_k by $P_{k+N_{\mathsf{voters}}}$ for each $k \in N_{\mathsf{voters}} + [N_{\mathsf{trustees}}]$. By Definition 5.1.1 there exists $n_0 \in \mathbb{N}$ such that for all input tuples x and for $\lambda \geq n_0$ it holds true that $\mathsf{Pr}[\mathsf{transcript} \notin \tilde{\gamma} \wedge J : \mathsf{accept}] \leq \delta$ over the probability space of $\tilde{\mathcal{P}}^*\left(1^\lambda, x\right)$ Let x be an input tuple and $\lambda \geq \max\{n_0, n_1\}$ for n_1 to be specified later. We will now prove that

$$\text{(5.1)} \quad \Pr_{\mathcal{P}^*\left(1^\lambda, x\right)}\Big[\underbrace{\mathsf{transcript}}_{=: T} \notin \gamma \ \wedge \ \underbrace{J : \mathsf{accept}}_{=: A_J}\Big] \leq \Pr_{\tilde{\mathcal{P}}^*\left(1^\lambda, x\right)}\Big[\Big(\underbrace{\mathsf{transcript}}_{=: \tilde{T}} \notin \tilde{\gamma} \ \vee \ \exists i \in [N_{\mathsf{voters}}] \setminus K' : \neg \tilde{A}_{\tilde{V}_i}\Big) \ \wedge \ \underbrace{J : \mathsf{accept}}_{=: \tilde{A}_J}\Big]$$

where the event $\tilde{A}_{\tilde{V}_i}$ describes that the judging procedure ran by the (honest) voter $\tilde{V}_i$ accepted.

Let r be random coins such that the left-hand event occurs; that is, $T(r) \notin \gamma$ and $r \in A_J$. In order for Inequality 5.1 to hold true it suffices to prove that with the same random coins the right-hand event occurs, that is, at least one of the following statements is true.

- $r \in \tilde{A}_J$.
- If for all $i \in [N_{\mathsf{voters}}] \setminus K'$ it holds true that $r \in \tilde{A}_{\tilde{V}_i}$, then $\tilde{T}(r) \in \tilde{\gamma}$.

Recall that the only difference between $\mathcal{P}^*$ and $\tilde{\mathcal{P}}^*$ are the honest voters and this difference is only in their behaviour after they sent their ballot. Therefore J does exactly the same in both protocol runs. In particular he accepts and thus $r \in \tilde{A}_J$. Now assume that for all $i \in [N_{\mathsf{voters}}] \setminus K'$ it holds true that

$r \in \tilde{A}_{\tilde{V}_i}$. That is, all of the judging procedures ran by the honest voters $\tilde{V}_i$ in $\tilde{\mathcal{P}}^*$ accept. An accepting judging procedure does not change the behaviour of $\tilde{V}_i$ relative to V_i and hence $T(r) = \tilde{T}(r)$ and $K((r)) = \tilde{K}\big(\tilde{T}(r)\big)$. With those two equalities the definitions of $T(r) \in \gamma$ and $\tilde{T}(r) \in \tilde{\gamma}$ coincide and therefore we conclude that $\tilde{T}(r) \notin \tilde{\gamma}$.

We just concluded the proof of Inequality 5.1. With negligible knowledge error there's a negligible probability that the judging procedure has distinct outputs when ran by J and $\tilde{V}_i$. It follows that

$$\begin{aligned}
\mathsf{Pr}[T \notin \gamma \wedge A_J] &\leq \mathsf{Pr}\Big[\Big(\tilde{T} \notin \gamma \vee \exists i \in [N_{\mathsf{voters}}] \setminus K' : \neg \tilde{A}_{\tilde{V}_i}\Big) \wedge \tilde{A}_J\Big] \\
&= \mathsf{Pr}\Big[\big(\tilde{T} \notin \gamma \wedge \tilde{A}_J\big) \vee \Big(\exists i \in [N_{\mathsf{voters}}] \setminus K' : \neg \tilde{A}_{\tilde{V}_i} \wedge \tilde{A}_J\Big)\Big] \\
&\leq \mathsf{Pr}\big[\big(\tilde{T} \notin \gamma \wedge \tilde{A}_J\big)\big] + \mathsf{Pr}\Big[\Big(\exists i \in [N_{\mathsf{voters}}] \setminus K' : \neg \tilde{A}_{\tilde{V}_i} \wedge \tilde{A}_J\Big)\Big] \\
&\leq \frac{\delta}{2} + \mathsf{negl}(\lambda) \\
&\leq \delta
\end{aligned}$$

if we choose n_1 large enaugh. □

5.4 Suitable Result Functions

Our E-voting system EV_f is of course only tally-hiding if the result function f is considered to be tally-hiding. In this section we construct suitable result functions. Recall that the function must be given as a 1 bounded arithmetic circuit, because it is evaluated homomorphically. For now, assume that there are only two candidates and that the result function is a threshold function with threshold $t \in [N_{\mathsf{voters}}]$. That is,

$$\begin{aligned}
f_1 : \quad & \{0,1\}^{N_{\mathsf{voters}}} \to \{0,1\} \\
& \big(\mathsf{ch}_1, \ldots, \mathsf{ch}_{N_{\mathsf{voters}}}\big) \mapsto \begin{cases} 0 & \text{if } \sum_{i=1}^{N_{\mathsf{voters}}} \mathsf{ch}_i < t \\ 1 & \text{if } \sum_{i=1}^{N_{\mathsf{voters}}} \mathsf{ch}_i \geq t. \end{cases}
\end{aligned} \tag{5.2}$$

This result function allows us to directly use the voter's choices as inputs to the circuit. The circuit itself can be implemented as follows. Each vote is interpreted as a 1-bit binary number. The votes are then added using carry ripple adders. This summation is done in a dovetailed fashion in order to keep the depth of the circuit small. There are $\lceil \log N_{\mathsf{voters}} \rceil$ levels of carry ripple adders. At each level the number of bits increases by 1, so the result is a $\lceil \log N_{\mathsf{voters}} \rceil + 1$ bit binary number. We extend this to a $\lceil \log N_{\mathsf{voters}} \rceil + 2$ bit two's complement binary number by prepending the sign bit 0. In order to subtract t, a virtual vote is assembled which is the $\lceil \log N_{\mathsf{voters}} \rceil + 2$ bit two's complement of the binary representation of t. This virtual vote is added to the previous result and the output then is one minus the sign bit of the overall sum. Let $l := \lceil \log N_{\mathsf{voters}} \rceil + 1$. The depth of this circuit is in $\Theta\big(l^2\big)$, leading to an Ring-GSW modulus q with $\log q \in O\big(l^2 \log l\big) = O\big(\log^2 N_{\mathsf{voters}} \log\log N_{\mathsf{voters}}\big)$ for fixed security parameter. The circuit has size $|f_1| \in \Theta\big(N_{\mathsf{voters}} \log N_{\mathsf{voters}}\big)$. Substituting this into the time complexity for variable circuits from Table 4.1, we conclude that the time complexity is in $\tilde{\Theta}\big(N_{\mathsf{voters}}\big)$, as $\tilde{\Theta}(\,\cdot\,)$ absorbs the logarithmic terms. Note that this is the time complexity for trustees and judges. Voters have quasi-constant (i.e. $\tilde{\Theta}\big(1\big)$) time complexity in the number of voters.

A careful analysis yields a better bound for the exponent of the polylogarithmic factor than above analysis which is merely based on the circuit's depth. We can implement a full adder as

$$\begin{aligned}&\{0,1\}^3 \mapsto \{0,1\}^2\\ &\begin{pmatrix} a \\ b \\ c_{\mathsf{in}} \end{pmatrix} \mapsto \begin{bmatrix} s = (1-2h)\cdot a + h \\ c_{\mathsf{out}} = c_{\mathsf{in}} \cdot (a+b) + (1-2c_{\mathsf{in}}) \cdot ab \end{bmatrix} \quad \text{with} \quad h = (1-2c_{\mathsf{in}}) \cdot b + c_{\mathsf{in}}.\end{aligned}$$

Let $\mathbf{A}, \mathbf{B}, \mathbf{C}_{\mathsf{in}}, \mathbf{S}, \mathbf{C}_{\mathsf{out}}$ be corresponding ciphertexts. When computing the noise of $\mathbf{S}$ and $\mathbf{C}_{\mathsf{out}}$ we obtain

$$\begin{aligned}||\mathsf{noise}(\mathbf{S})||_\infty &\leq ||(1-2(a\oplus b))\mathsf{noise}(\mathbf{C}_{\mathsf{in}})||_\infty \\ &\quad + m||\mathsf{noise}(\mathbf{A})||_\infty + m||(1-2a)\mathsf{noise}(\mathbf{B})||_\infty \\ ||\mathsf{noise}(\mathbf{C}_{\mathsf{out}})||_\infty &\leq ||(a\oplus b)\,\mathsf{noise}(\mathbf{C}_{\mathsf{in}})||_\infty \\ &\quad + m||(1+b)\,\mathsf{noise}(\mathbf{A})||_\infty + \Big(m^2+m\Big)||\mathsf{noise}(\mathbf{B})||_\infty.\end{aligned} \tag{5.3}$$

Observe that $1-2(a\oplus b), 1-2a \in \{-1,1\}$, so these factors can be dropped. Let $N_{\mathbf{A},\mathbf{B}} := \max\{||\mathsf{noise}(\mathbf{A})||_\infty, ||\mathsf{noise}(\mathbf{B})||_\infty\}$. It follows

$$\begin{aligned}||\mathsf{noise}(\mathbf{S})||_\infty &\leq ||\mathsf{noise}(\mathbf{C}_{\mathsf{in}})||_\infty + 2mN_{\mathbf{A},\mathbf{B}} \\ ||\mathsf{noise}(\mathbf{C}_{\mathsf{out}})||_\infty &\leq ||\mathsf{noise}(\mathbf{C}_{\mathsf{in}})||_\infty + \Big(m^2+3m\Big)N_{\mathbf{A},\mathbf{B}}.\end{aligned}$$

Interestingly, when computing c_{out} this way, then the noise level of $\mathbf{C}_{\mathsf{out}}$ does only depend on the noise level $\mathbf{C}_{\mathsf{in}}$ by a factor of 1. Therefore the noise level of the output of a carry ripple adder is only linear in the number of bits, while with any factor > 1, this noise level would be exponential in the number of bits.

In order to analyze the dovetailed sum we need some notation. Let $\mathbf{C}_{0,0} = \mathbf{S}_{0,0}$ be the input ciphertext with maximum noise level. For $i \in \{1,\ldots,\lceil \log N_{\mathsf{voters}}\rceil + 1\}$ and $j \in \{0,\ldots,i\}$, let $\mathbf{S}_{i,j}$ be the maximum noise level ciphertext corresponding to a 2^j sum bit in the i^{th} layer of carry ripple adders. And let $\mathbf{C}_{i,j}$ be the maximum noise level ciphertext corresponding to a 2^j carry-in or equivalently a 2^{j-1} carry-out bit in the i^{th} layer of carry ripple adders. Note that $\mathbf{C}_{i,i} = \mathbf{S}_{i,i}$ because the last carry-out bit is at the same time the last sum bit. Then we obtain the system of recurrence inequalities

$$\begin{aligned}&\big\|\mathsf{noise}\big(\mathbf{S}_{i,j}\big)\big\|_\infty \leq \big\|\mathsf{noise}\big(\mathbf{C}_{i,j}\big)\big\|_\infty + 2m\big\|\mathsf{noise}\big(\mathbf{S}_{i-1,j}\big)\big\|_\infty && \forall i > j \geq 0\\ &\big\|\mathsf{noise}\big(\mathbf{C}_{i,0}\big)\big\|_\infty = 0 && \forall i > 0\\ &\big\|\mathsf{noise}\big(\mathbf{C}_{i,j}\big)\big\|_\infty \leq \big\|\mathsf{noise}\big(\mathbf{C}_{i,j-1}\big)\big\|_\infty + \Big(m^2+3m\Big)\big\|\mathsf{noise}\big(\mathbf{S}_{i-1,j-1}\big)\big\|_\infty && \forall i \geq j > 0.\end{aligned}$$

Define $B_0 := \big\|\mathsf{noise}\big(\mathbf{C}_{0,0}\big)\big\|_\infty$ and recall that $l = \lceil \log N_{\mathsf{voters}}\rceil + 1$. By induction it can be proven that

$$\big\|\mathsf{noise}\big(\mathbf{S}_{i,j-1}\big)\big\|_\infty, \big\|\mathsf{noise}\big(\mathbf{C}_{i,j}\big)\big\|_\infty \leq (i-1)!j \cdot \Big(m^2+3m\Big)^i B_0 \quad \forall i \geq j > 0.$$

Of all ciphertexts, the most loose bound is obtained for $\mathbf{C}_{l,l} = \mathbf{S}_{l,l}$.

We did not yet take account of the virtual vote which is the two's complement of t. The virtual vote itself has no noise as it's based solely on public information. Therefore, when adding the virtual vote using another carry ripple adder, one of the terms from Equation (5.3) vanishes and consequently the noise increases only by an additional factor of ml.

For the final noise level, hereafter denoted by B_{noise}, we obtain

$$\log B_{\mathsf{noise}} \leq \log\left(ml \cdot l! \cdot \left(m^2 + 3m\right)^l B_0\right) \in O\big(l \log l\big)$$

for constant m. However, $m = 2(\lfloor \log q \rfloor + 1)$ is not constant but rather depends on the Ring-GSW modulus q which itself must be chosen in the order of $q \in \Omega\big(B_{\mathsf{noise}}\big)$. Taking this into account does not change the result of $\log B_{\mathsf{noise}} \in O\big(l \log l\big)$. Therefore we can choose $\log q \in O\big(l \log l\big) = O\big(\log N_{\mathsf{voters}} \log\log N_{\mathsf{voters}}\big)$ for fixed security parameter. We still have time complexity in $\tilde{\Theta}\big(N_{\mathsf{voters}}\big)$, but the exponent of the polylogarithmic term is now only about half as large as with our previous result that was merely based on the depth of the circuit.

Often there are more than two candidates. Let's consider any candidate set $\mathsf{Ch} = [C]$ with $C \in \mathbb{N}$. We will now implement the ordinary majority result function

$$\begin{aligned} f_2 : \quad & [C]^{N_{\mathsf{voters}}} \to [C] \cup \{\bot\} \\ & \big(\mathsf{ch}_1, \ldots, \mathsf{ch}_{N_{\mathsf{voters}}}\big) \mapsto \begin{cases} c & \text{if } |\{i \in [N_{\mathsf{voters}}] \mid \mathsf{ch}_i = c\}| > N_{\mathsf{voters}}/2 \\ \bot & \text{else} \end{cases} \end{aligned} \tag{5.4}$$

as a 1-bounded arithmetic circuit. Elements of $\mathsf{Ch} = [C]$ can not be used as an input to the circuit. Therefore the voters have to perform some preprocessing. The most simple and straightforward way is to give the binary representation of ch_i as an input to the circuit. Let's assume for brevity that C is a power of two, although other integers are possible via a simple modification. The binary representation of ch_i has $\log|\mathsf{Ch}|$ bits. In the circuit the representation is then converted to unary by computing a conjunction (product) of $\log|\mathsf{Ch}|$ literals for each unary bit. The asymmetry of

$$\mathsf{noise}(\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2)) = \mu_2 \mathsf{noise}(\mathbf{C}_1) + \mathbf{G}^{-1}(\mathbf{C}_1)\,\mathsf{noise}(\mathbf{C}_2),$$

from Equation (3.3) implies that, in the *left associative* multiplication of several ciphertexts, the noise grows only linearly rather than exponentially in the number of multiplications. The conversion from binary to unary increases the noise level by a factor of at most $m \log|\mathsf{Ch}|$. Each bit of the unary representation corresponds to one candidate. Subsequently for each candidate $c \in \mathsf{Ch}$ a copy of the circuit which computes Equation (5.2) with $t = \lfloor N_{\mathsf{voters}}/2 \rfloor + 1$ is run on the bits corresponding to that candidate. The output of f_2 are the (in total) $|C|$ output bits of the $|C|$ sub-circuits. Notice that at most one of these bits can be 1 because there can not be multiple candidates that have the ordinary majority. The conversion from binary to unary where the noise level increases by a factor of up to $m \log|\mathsf{Ch}|$ changes our asymptotic result slightly to $\log q \in O\big(\log N_{\mathsf{voters}} \log\log N_{\mathsf{voters}}\big) + O\big(\log\log|\mathsf{Ch}|\big)$.

The last result function which we mention here is for the case where one wants to order the candidates by their number of votes. We do so by outputting for each pair (c_j, c_k) of candidates whether c_j got less, equal or more votes than c_k. Define

$$\begin{aligned} f_3 : \quad & [C]^{N_{\mathsf{voters}}} \to \{-1, 0, 1\}^{C \times C} \\ & \big(\mathsf{ch}_1, \ldots, \mathsf{ch}_{N_{\mathsf{voters}}}\big) \mapsto \big(b_{j,k}\big)_{j,k \in [C]} \end{aligned} \tag{5.5}$$

Circuit	Maximum N_{voters} using parameters from Table 3.1
f_1	256
f_2 (assuming $\lvert\mathsf{Ch}\rvert \leq N_{\mathsf{voters}}$)	128
f_3 (assuming $\lvert\mathsf{Ch}\rvert \leq N_{\mathsf{voters}}$)	128

Table 5.1: Comparison of maximum N_{voters}, assuming $N_{\mathsf{trustees}} \leq 4096$

Circuit	Number of gates
f_1	$\Theta\big(N_{\mathsf{voters}} \log N_{\mathsf{voters}}\big)$
f_2	$\Theta\big(\lvert\mathsf{Ch}\rvert \log\lvert\mathsf{Ch}\rvert + \lvert\mathsf{Ch}\rvert N_{\mathsf{voters}} \log N_{\mathsf{voters}}\big)$
f_3	$\Theta\big(\lvert\mathsf{Ch}\rvert^2 + \lvert\mathsf{Ch}\rvert N_{\mathsf{voters}} \log N_{\mathsf{voters}}\big)$

Table 5.2: Circuit complexity comparison

with

$$b_{j,k} = \operatorname{sign}\Big(\Big|\big\{i \in [N_{\mathsf{voters}}] \;\Big|\; \mathsf{ch}_i = c_j\big\}\Big| - |\{i \in [N_{\mathsf{voters}}] \mid \mathsf{ch}_i = c_k\}|\Big).$$

We can implement this by first computing the number of votes for each candidate as before. Afterwards, for each pair c_j, c_k compute the difference (using two's complement) and denote the sign bit of the difference by $s_{j,k}$. Then $b_{j,k} = s_{k,j} - s_{j,k}$.

In Table 5.1 we list the maximum values for N_{voters} when homomorphically evaluating one of the constructed circuits using one of our parameter sets from Table 3.1. Note that it does not matter which of two parameter sets is used because they both have the same ratio $q/2^\lambda$ which is the maximum allowed noise level. The choice of the parameter set is a choice merely concerning the level of security. The table assumes $\lvert\mathsf{Ch}\rvert \leq N_{\mathsf{voters}}$ which was an arbitrary assumption in order to compute the bounds. Of course there can be more choices than voters, but then the maximum number of voters (using the same parameter set) might be slightly lower. The same goes for the assumption $N_{\mathsf{trustees}} \leq 4096$.

Table 5.2 shows the asymptotic number of gates for the circuits we constructed. Each gate corresponds to a homomorphic operation and each homomorphic operation takes time polynomial in l and thus polylogarithmic in N_{voters}. Hence the overall runtime of the homomorphic evaluation is the same as the number of gates up to a polylogarithmic factor.

5.5 Result Function Computation over the Symmetric Group

In Section 3.1 we outlined that the Ring-GSW homomorphic encryption scheme has acceptable noise growth when evaluating an arbitrary 1-bounded arithmetic circuit. Every efficiently computable function can — for given input size — be transformed into such an arithmetic circuit. However, for specific functions there might be an algebraic (and thus maybe conceptually simpler or more natural) implementation. We give such an implementation for functions that use only additions and range checks over $\mathbb{Z}_n$ (for some $n \in \mathbb{N}$).

In order to analyze noise growth, we have already made use of the fact that in a left associative product the noise grows only linear in the number of multiplications. The same follows for the product of several permutation matrices [AP14; BV13]. This motivates to perform computations in the symmetric group S_n (for n to be specified later).

Let $\varphi(1) := (j \mapsto j+1 \mod n)$. We define the homomorphism $\varphi : \mathbb{Z}_n \to S_n$ by $\varphi(i) = \varphi(1)^i$ for $i \in \mathbb{Z}_n$. It is easy to see that φ is injective. We represent each element $\pi \in S_n$ as permutation matrix $\pi = \left(1_{i=\pi(j)}\right)_{i,j\in\mathbb{Z}_n}$. In such a permutation matrix, each column and each row has exactly one 1-entry. Observe that, for $i \in \mathbb{Z}_n$, $\varphi(i)$ is the permutation matrix that has a 1 in the i^{th} row of the first column and where every further column is the previous column rotated by one. Hence an admissible permutation matrix is sufficiently specified by its first column. If a voter V_i wants to input a value $i \in \mathbb{Z}_n$, then V_i instead inputs the first column of $\varphi(i)$ in encrypted form. If that input is restricted to some values from $\mathbb{Z}_n$, then V_i only provides the corresponding entries of the first column.

An addition $i + j$ in $\mathbb{Z}_n$ can be performed by instead multiplying the permutation matrices $\varphi(i)\varphi(j) = \varphi(i+j)$. Furthermore, for any subset $I \subseteq \mathbb{Z}_n$, we can check whether $i \in I$ by adding the corresponding entries in the first column of $\varphi(i)$. Formally that is

$$i \in I \quad \Longleftrightarrow \quad \sum_{j\in I} \varphi(i)\,[1, j+1] = 1.$$

In order to maintain security against active adversaries, a voter V_i that inputs a column of a permutation matrix in encrypted form has to prove that the column contains exactly one 1. The ZKPoK that already certifies correct encryption can be adapted in order to account for this. Alternatively, one can consider a result function where V_i inputs only a single entry of the permutation matrix and hence cannot cheat. The function f_1 from Equation (5.2) is such a case.

Computing f_1 in S_n works as follows. We use $n := N_{\text{voters}}+1$ in order to prevent a wrap-around. Each voter V_i with choice $\mathsf{ch}_i \in \{0, 1\}$ inputs $\varphi(\mathsf{ch}_i)$. Note that V_i actually only has to input the top left entry $\varphi(i)\,[1, 1]$ (representing a 0), the subjacent entry is computed as $\varphi(i)\,[2, 1] = 1 - \varphi(i)\,[1, 1]$ and all other entries of the first column are filled by zeros. When each voter has input her permutation matrix, then these are multiplied in order to obtain $\mathbf{P} := \prod_i \varphi(\mathsf{ch}_i) = \varphi(\sum_i \mathsf{ch}_i \mod n)$. The result is then obtained from a range check

$$\sum_i \mathsf{ch}_i \geq t \quad \Longleftrightarrow \quad \sum_{j=t}^{N_{\text{voters}}} \mathbf{P}[1, j+1].$$

As we mentioned above, the noise grows only linear in k when multiplying k permutation matrices. Specifically, the noise level increases by a factor of kmN where N is the width of the permutation matrix. In our cases $k = N = N_{\text{voters}}$, so that's a factor of $m{N_{\text{voters}}}^2$. Together with the subsequent sum, the total noise growth of $m{N_{\text{voters}}}^3$ is still cubic and leads to an Ring-GSW modulus $q \in O\left({N_{\text{voters}}}^3\right)$. This is in contrast to the superpolynomial (in N_{voters}) noise growth when computing f_1 using carry ripple adders. Let $l := \lceil \log N_{\text{voters}} \rceil + 1$ as before. In terms of l, it follows that we can choose $\log q \in O\left(\log\left({N_{\text{voters}}}^3\right)\right) = O\left(\log\left(2^{3l}\right)\right) = O(l)$. This is in contrast to our previous results where $\log q \in O(l \log l)$ was only quasi-linear in l. It comes at the cost that we have to do a lot of matrix operations. Since we only need the first column of a permutation matrix, each matrix-matrix product can be reduced to a matrix-vector product. Nevertheless we need $\Theta\left({N_{\text{voters}}}^3\right)$ homomorphic

operations for N_{voters} such products. Each homomorphic operation takes time polylogarithmic in N_{voters}, so the overall runtime is in $\tilde{\Theta}\left({N_{\text{voters}}}^3\right)$. This is worse than our previous result of $\tilde{\Theta}\left(N_{\text{voters}}\right)$ (in Section 5.4) and thus we deem the computation using carry ripple adders better, both practically and asymptotically.

6 Conclusion and Outlook

We constructed a levelled FHE scheme based on the Ring-LWE assumption. This scheme is the Ring-LWE analogous to the GSW FHE scheme which is based on standard LWE. Using the key homomorphic property of the scheme, we extended it to a *threshold* levelled FHE scheme by augmenting a distributed key generation and distributed encryption. This followed the example set in [AJW11] where the BGV scheme was used instead. Our threshold scheme led to a passively secure 4-round MPC protocol in the common random string model.

By augmenting Fiat-Shamir transformed PLS proofs by del Pino et al. [PLS19], we obtained an actively secure 4-round MPC protocol. The security of this protocol is based on the additional assumption that the discrete-log problem in the used group is hard. Even if discrete-log will be broken in the future (by e.g. practicable quantum computers), the privacy of past protocol executions is still preserved. The space, time and communication complexity of both of our protocols are polynomial in the number of parties, in the security parameter and in the depth of the arithmetic circuit. However, the constant exponents are quite large except in the dependence on the number of parties.

Based on our MPC protocol we finally constructed an end-to-end verifiable E-voting system in the ROM which can be instantiated with arbitrary result functions. For three families of tally-hiding result functions we constructed the corresponding families of arithmetic circuits and analyzed the asymptotic complexity of the resulting tally-hiding E-voting system. Interestingly we obtained time complexity in $\tilde{\Theta}(N_{\text{voters}})$ which is optimal up to a polylogarithmic factor.

An experimental implementation of ActiveMPC which also supports usage as an E-voting system as described in Section 5.3 will be available at [Has20]. We conclude the thesis by providing an outlook on possible future work.

6.1 Threshold Levelled FHE without Smudging

In Section 3.5 when choosing q we had to multiply a factor of 2^λ in order for the smudging to be secure. In practice this has an effect that is not to be underestimated: If the factor of 2^λ in q was not necessary, then our public key in Table 3.1 for $\lambda = 128$ (resp. $\lambda = 256$) would have size ≈ 528 MiB instead of ≈ 1.67 GiB (resp. ≈ 3.59 GiB instead of ≈ 7.94 GiB). As stated in Section 3.5 we leave it as an open question whether it is possible to eliminate smudging from the protocol.

Bibliography

[ABC+15] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, M. Strand. "A Guide to Fully Homomorphic Encryption". In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1192. URL: http://eprint.iacr.org/2015/1192 (cit. on pp. 9, 17).

[ACPS09] B. Applebaum, D. Cash, C. Peikert, A. Sahai. "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems". In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Ed. by S. Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 595–618. DOI: 10.1007/978-3-642-03356-8_35. URL: https://doi.org/10.1007/978-3-642-03356-8_35 (cit. on p. 21).

[AJW11] G. Asharov, A. Jain, D. Wichs. "Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE". In: *IACR Cryptology ePrint Archive* 2011 (2011), p. 613. URL: http://eprint.iacr.org/2011/613 (cit. on pp. 9, 29, 30, 41, 65).

[AP14] J. Alperin-Sheriff, C. Peikert. "Faster Bootstrapping with Polynomial Error". In: *IACR Cryptology ePrint Archive* 2014 (2014), p. 94. URL: http://eprint.iacr.org/2014/094 (cit. on pp. 23, 24, 62).

[BCK+14] F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, G. Neven. "Better Zero-Knowledge Proofs for Lattice Encryption and Their Application to Group Signatures". In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*. Ed. by P. Sarkar, T. Iwata. Vol. 8873. Lecture Notes in Computer Science. Springer, 2014, pp. 551–572. DOI: 10.1007/978-3-662-45611-8_29. URL: https://doi.org/10.1007/978-3-662-45611-8_29 (cit. on p. 48).

[BGR+15] H. Brenner, V. Goyal, S. Richelson, A. Rosen, M. Vald. "Fast Non-Malleable Commitments". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*. Ed. by I. Ray, N. Li, C. Kruegel. ACM, 2015, pp. 1048–1057. DOI: 10.1145/2810103.2813721. URL: https://doi.org/10.1145/2810103.2813721 (cit. on p. 13).

[BGV11] Z. Brakerski, C. Gentry, V. Vaikuntanathan. "Fully Homomorphic Encryption without Bootstrapping". In: *IACR Cryptology ePrint Archive* 2011 (2011), p. 277. URL: http://eprint.iacr.org/2011/277 (cit. on pp. 9, 22, 24, 29, 41).

[BR93] M. Bellare, P. Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”. In: *CCS ’93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*. Ed. by D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, V. Ashby. ACM, 1993, pp. 62–73. DOI: 10.1145/168588.168596. URL: https://doi.org/10.1145/168588.168596 (cit. on pp. 10, 11).

[BV13] Z. Brakerski, V. Vaikuntanathan. “Lattice-Based FHE as Secure as PKE”. In: *IACR Cryptol. ePrint Arch.* 2013 (2013), p. 541. URL: http://eprint.iacr.org/2013/541 (cit. on p. 62).

[FF00] M. Fischlin, R. Fischlin. “Efficient Non-malleable Commitment Schemes”. In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. Ed. by M. Bellare. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 413–431. DOI: 10.1007/3-540-44598-6_26. URL: https://doi.org/10.1007/3-540-44598-6_26 (cit. on pp. 9, 11).

[FS86] A. Fiat, A. Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *Advances in Cryptology - CRYPTO ’86, Santa Barbara, California, USA, 1986, Proceedings*. Ed. by A. M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194. DOI: 10.1007/3-540-47721-7_12. URL: https://doi.org/10.1007/3-540-47721-7_12 (cit. on pp. 9, 49).

[GPR15] V. Goyal, O. Pandey, S. Richelson. “Textbook Non-Malleable Commitments”. In: *IACR Cryptol. ePrint Arch.* 2015 (2015), p. 1178. URL: http://eprint.iacr.org/2015/1178 (cit. on p. 13).

[GPV07] C. Gentry, C. Peikert, V. Vaikuntanathan. “Trapdoors for Hard Lattices and New Cryptographic Constructions”. In: *IACR Cryptology ePrint Archive* 2007 (2007), p. 432. URL: http://eprint.iacr.org/2007/432 (cit. on p. 45).

[GSW13] C. Gentry, A. Sahai, B. Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: *IACR Cryptology ePrint Archive* 2013 (2013), p. 340. URL: http://eprint.iacr.org/2013/340 (cit. on pp. 9, 23, 24, 41).

[Has20] S. Hasler. *RGSW Multi-Party Computation*. 2020. URL: https://github.com/haslersn/rgsw (cit. on pp. 10, 65).

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, M. Luby. “A Pseudorandom Generator from any One-way Function”. In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396. DOI: 10.1137/S0097539793244708. URL: https://doi.org/10.1137/S0097539793244708 (cit. on p. 24).

[ILL89] R. Impagliazzo, L. A. Levin, M. Luby. “Pseudo-random Generation from one-way functions (Extended Abstracts)”. In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*. Ed. by D. S. Johnson. ACM, 1989, pp. 12–24. DOI: 10.1145/73007.73009. URL: https://doi.org/10.1145/73007.73009 (cit. on p. 24).

[KTV10] R. Küsters, T. Truderung, A. Vogt. “Accountability: Definition and Relationship to Verifiability”. In: *IACR Cryptology ePrint Archive* 2010 (2010), p. 236. URL: http://eprint.iacr.org/2010/236 (cit. on pp. 16, 54).

[KW11] R. Küsters, T. Wilke. *Moderne Kryptographie - Eine Einführung*. Vieweg + Teubner, 2011. ISBN: 978-3-519-00509-4 (cit. on pp. 12, 15).

[LNSW12] S. Ling, K. Nguyen, D. Stehlé, H. Wang. "Improved Zero-knowledge Proofs of Knowledge for the ISIS Problem, and Applications". In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 569. URL: http://eprint.iacr.org/2012/569 (cit. on p. 48).

[LPR12] V. Lyubashevsky, C. Peikert, O. Regev. "On Ideal Lattices and Learning with Errors Over Rings". In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 230. URL: http://eprint.iacr.org/2012/230 (cit. on pp. 9, 19–21, 24, 45).

[LPR13] V. Lyubashevsky, C. Peikert, O. Regev. *A Toolkit for Ring-LWE Cryptography*. Cryptology ePrint Archive, Report 2013/293. https://eprint.iacr.org/2013/293. 2013 (cit. on pp. 19, 20).

[Mül19] J. Müller. "Design and Cryptographic Security Analysis of E-Voting Protocols". PhD thesis. Institut für Informationssicherheit der Universit¨at Stuttgart, 2019 (cit. on pp. 9, 53, 54).

[PLS19] R. del Pino, V. Lyubashevsky, G. Seiler. "Short Discrete Log Proofs for FHE and Ring-LWE Ciphertexts". In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 57. URL: https://eprint.iacr.org/2019/057 (cit. on pp. 9, 45, 46, 48, 49, 51, 65).

[PS96] D. Pointcheval, J. Stern. "Security Proofs for Signature Schemes". In: *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*. Ed. by U. M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 387–398. DOI: 10.1007/3-540-68339-9_33. URL: https://doi.org/10.1007/3-540-68339-9_33 (cit. on p. 49).

[Reg05] O. Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. Ed. by H. N. Gabow, R. Fagin. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603 (cit. on pp. 9, 24).

[Sch80] J. T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities". In: *J. ACM* 27.4 (1980), pp. 701–717. DOI: 10.1145/322217.322225. URL: https://doi.org/10.1145/322217.322225 (cit. on p. 48).

[SS71] A. Schönhage, V. Strassen. "Schnelle Multiplikation großer Zahlen". In: *Computing* 7.3-4 (1971), pp. 281–292. DOI: 10.1007/BF02242355. URL: https://doi.org/10.1007/BF02242355 (cit. on p. 43).

[Vad12] S. P. Vadhan. "Pseudorandomness". In: *Foundations and Trends in Theoretical Computer Science* 7.1-3 (2012), pp. 1–336. DOI: 10.1561/0400000010. URL: https://doi.org/10.1561/0400000010 (cit. on p. 12).

[Zip79] R. Zippel. "Probabilistic algorithms for sparse polynomials". In: *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*. Ed. by E. W. Ng. Vol. 72. Lecture Notes in Computer Science. Springer, 1979, pp. 216–226. DOI: 10.1007/3-540-09519-5_73. URL: https://doi.org/10.1007/3-540-09519-5_73 (cit. on p. 48).

All links were last followed on May 14, 2020.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Stuttgart, 14.5.2020, S. Hark

place, date, signature