Zero-Knowledge Proofs with Low Amortized Communication from Lattice Assumptions

Ivan Damgård[*]
Aarhus University

Adriana López-Alt
New York University

June 26, 2012

Abstract

We construct zero-knowledge proofs of plaintext knowledge (PoPK) and correct multiplication (PoPC) for the Regev encryption scheme with low amortized communication complexity. Previous constructions of both PoPK and PoPC had communication cost linear in the size of the public key (roughly quadratic in the lattice dimension, ignoring logarithmic factors). Furthermore, previous constructions of PoPK suffered from one of the following weaknesses: either the message and randomness space were restricted, or there was a *super-polynomial* gap between the size of the message and randomness that an honest prover chose and the size of which an accepting verifier would be convinced. The latter weakness was also present in the existent PoPC protocols.

In contrast, $O(n)$ proofs (for lattice dimension n) in our PoPK and PoPC protocols have communication cost linear in the public key. Thus, we improve the *amortized* communication cost of each proof by a factor linear in the lattice dimension. Furthermore, we allow the message space to be $\mathbb{Z}_p$ and the randomness distribution to be the discrete Gaussian, both of which are natural choices for the Regev encryption scheme. Finally, in our schemes there is no gap between the size of the message and randomness that an honest prover chooses and the size of which an accepting verifier is convinced.

Our constructions use the "MPC-in-the-head" technique of Ishai et al. (STOC 2007). At the heart of our constructions is a protocol for proving that a value is bounded by some publicly known bound. This uses Lagrange's Theorem that states that any positive integer can be expressed as the sum of four squares (an idea previously used by Boudot (EUROCRYPT 2000)), as well as techniques from Cramer and Damgård (CRYPTO 2009).

[*]The first author acknowledges support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.

1 Introduction

The problem of secure multiparty computation (MPC) [GMW87, BGW88, CCD88, Yao82] is central in the field of modern cryptography. In this problem, N parties $\mathcal{P}_1, \ldots, \mathcal{P}_N$ holding private inputs $x_1, \ldots, x_N$, respectively, wish to compute a function $f(x_1, \ldots, x_N)$ on their inputs without revealing any information apart from the output of the evaluation (in particular, they wish to keep their inputs secret from the other parties). Solutions to this problem abound in the literature. Many of these solutions use the circuit rerandomization technique of Beaver [Bea91] (see e.g. [HM01, KK07, BH08, HNP08, DO10, BTHN10, BDOZ11, DPSZ11], among many others). Circuit rerandomization requires players to hold (additive) secret sharings of many random triples (a, b, c) such that $c = a \cdot b$ in some finite field. Traditionally, these triples are created using zero-knowledge proofs.

Bendlin et al. [BDOZ11] use zero-knowledge proofs of plaintext knowledge (PoPK) and correct multiplication (PoCM) for this purpose. To see how this is done, consider the 2-party setting as an example. To obtain an additive secret sharing of random values a, b, players $\mathcal{P}_1$ and $\mathcal{P}_2$ can each choose random values u_1, v_1 and u_2, v_2, respectively, and define $a = u_1 + u_2$ and $b = v_1 + v_2$. Obtaining an additive secret sharing of $c = a \cdot b$ is more involved. First, notice that $c = a \cdot b = (u_1 + u_2) \cdot (v_1 + v_2) = u_1v_1 + u_1v_2 + u_2v_1 + u_2v_2$. If $\mathcal{P}_1$ and $\mathcal{P}_2$ could obtain an additive sharing of each product $u_iv_j = y_{ij} + z_{ij}$ then they could obtain a sharing for c by simply adding each of these shares: $c = (y_{11} + y_{12} + y_{21} + y_{22}) + (z_{11} + z_{12} + z_{21} + z_{22})$. Thus, the problem reduces to having $\mathcal{P}_1$ and $\mathcal{P}_2$ obtain an additive sharing of the product of their inputs m_1 and m_2, respectively (in this case u_i and v_j).

This can be done with the following protocol. $\mathcal{P}_1$ encrypts his input under his public key pk and obtains a ciphertext $c_1 = \texttt{Enc}_{pk}(m_1; r_1)$, which he sends to $\mathcal{P}_2$. Upon receiving c_1, $\mathcal{P}_2$ computes a ciphertext $c_x = \texttt{Enc}_{pk}(x; r_x)$ of a random plaintext x and computes $c_2 = m_2 \cdot c_1 + c_x$, sends it to $\mathcal{P}_1$, and outputs $-x$ as his share. If the encryption scheme has certain homomorphic properties, then $c_2 = \texttt{Enc}_{pk}(m_1m_2 + x)$. $\mathcal{P}_1$ decrypts c_2 and outputs $m_1m_2 + x$ as his share, thus obtaining an additive sharing of m_1m_2.

However, when players are malicious, $\mathcal{P}_2$ needs to ensure that c_1 is a valid ciphertext and $\mathcal{P}_1$ needs to ensure that $\mathcal{P}_2$ performed the multiplication step correctly. This can be done by having $\mathcal{P}_1$ and $\mathcal{P}_2$ provide zero-knowledge proofs that they performed their respective operations correctly: $\mathcal{P}_1$ sends a *proof of plaintext knowledge*, proving that there exist m_1, r_1 such that $c_1 = \texttt{Enc}_{pk}(m_1; r_1)$, and $\mathcal{P}_2$ sends a *proof of correct multiplication*, proving that there exist m_2, x, r_x such that $c_2 = m_2 \cdot c_1 + \texttt{Enc}_{pk}(x; r_x)$.

Unfortunately, these zero-knowledge proofs can incur a large communication cost, which increases the overall communication complexity of the MPC protocol in which they are used. A key observation is that even though many triples need to be created, they can be created simultaneously. This leads to the question of whether we can lower the *amortized* communication complexity of each proof, thus lowering the *total* communcation cost of all proofs. In this work, we answer this question affirmatively when the encryption scheme used is the Regev encryption scheme [Reg05], whose security is based on the hardness of the Learning with Errors (LWE) problem.

Related Work. Bendlin et al. [BDOZ11], Bendlin and Damgård [BD10], and Asharov et al. [AJW11, AJLA$^+$12] give constructions of proofs of plaintext knowledge. The work of [BDOZ11] shows proofs of plaintext knowledge for any "semi-homomorphic" encryption scheme, an example of which is the Regev scheme. When applied to this scheme, the communication cost of *each* proof is linear in the size of the public key (roughly quadratic in the lattice dimension, ignoring logarithmic factors). The works of [BD10] and [AJW11, AJLA$^+$12] show proofs of plaintext knowledge specifically for the Regev scheme, but here again, the communication cost of each proof is linear in the size of the public key. Similarly, [BDOZ11] shows proofs of correct multiplication which, when applied to the Regev encryption scheme, have communication complexity linear in the public key size per proof.

Unfortunately, the protocol of [BD10] only works for message space $\{0,1\}$ and randomness in $\{0,1\}^m$. Furthermore, the proofs of [BDOZ11] and [AJW11, AJLA$^+$12] suffer from the following weakness. To guarantee zero-knowledge, an honest prover must choose the message and randomness from a sufficiently small range. But in order to guarantee soundness against a cheating prover, we can only guarantee that if the verifier accepts then the message and randomness come from a much larger interval. Thus, there is a gap between the size of the witness of an honest prover and the size of which an accepting verifier will be convinced. Such a gap, which turns out to be *super-polynomial* in the security parameter, is undesirable.

Our Results and Techniques. We improve upon these results by showing proofs of plaintext knowledge and correct multiplication where the cost of $O(n)$ proofs, where n is the lattice dimension, is linear in the public key size. Thus, we improve the amortized cost of each proof by a linear factor in the lattice dimension. Furthermore, our protocol does not suffer from the weakness of [BDOZ11] and [AJW11, AJLA$^+$12]; there is no gap between the size of the witness of an honest prover and the size of which an accepting verifier is convinced. The message space in our schemes can be $\mathbb{Z}_p$ and the probability distribution for the randomness can be the discrete Gaussian.[1]

Our proof system uses the "MPC-in-the-head" technique of Ishai et al. [IKOS07], who show how to construct zero-knowlege proofs from MPC protocols. The basic idea is as follows. For an NP relation $R(x, w)$ with statement x and witness w, the prover runs an MPC protocol for the function $f_x(w) = R(x, w)$ "in his head" and commits to the view of each of the players. The verifier then outputs a subset T of the players as challenge, and the prover opens the commitments to the views of the players in T. If the views are consistent, the verifier accepts.

This is the same technique that was used in [BD10] yet we improve upon it. First, we also show how to obtain proofs of correct multiplication. But more importantly, we expand the proofs to allow the message space to be $\mathbb{Z}_p$ (rather than bits), and allow the randomness distribution to be the discrete Gaussian (rather than bit-vectors). To achieve this, we show a protocol that allows a dealer to prove that the secret that he secret-shared among N players is bounded by some publicly known bound B. The intuition behind this proof is as follows. Let $[s]$ denote the sharing of secret s. The dealer distributes a sharing of B, $[B]$, and the players compute sharings $[B - s]$ and $[B + s]$ by locally adding their corresponding shares. We know that $-B < s < B$ if and only if both $B - s$ and $B + s$ are positive, so the problem of proving that s is bounded by B reduces to proving that a secret s' that has been secret shared among N players is positive.

For this, we use Lagrange's Theorem that states that any positive integer can be written as the sum of four squares (see, e.g. [FR06]), and moreover, that these four squares can be computed efficiently [RS86, Lip03] (a similar technique was used by Boudot [Bou00]). The dealer computes u, v, w, y such that $s' = u^2 + v^2 + w^2 + y^2$, and distributes sharings $[u], [v], [w], [y]$. The players can then locally compute shares $[u^2 + v^2 + w^2 + y^2 - s'] = [0]$, and verify that these final shares

[1]Technically, we'll need the Regev scheme to have perfect correctness, so the randomness distribution will be a "truncated" discrete Gaussian that is statistically close to the discrete Gaussian, where values output according to the distribution are guaranteed to be small (as opposed to small with high probability).

reconstruct to 0.

However, we must ensure that the values u, v, w, y are all smaller than $\sqrt{q/8}$. Otherwise we can have overflow modulo q when we square and add the four squares, which would mean that we can no longer guarantee that the sum of the four squares is positive. For this, we use techniques from Cramer and Damgård [CD09]. The same techniques were used in [BDOZ11], yet the key difference is that we use them to bound the numbers to be squared (and thus the bound can be loose), whereas in [BDOZ11] they were used to bound the secrets themselves (thus leading to the gap discussed above). The use of this technique requires our modulus q to be super-polynomial in the security parameter λ (as was also the case in [BD10, BDOZ11, AJW11, AJLA+12]). See Section 3 for more details.

Other Applications. Recently, Brakerski et al. showed that a variant of the Regev scheme is fully homomorphic [BV11, BGV12]. The zero-knowledge PoPKs shown in this work can be used to prove that a ciphertext encrypted under this Regev-based FHE scheme is well-formed.

Presentation. In Section 2, we review some background needed for our constructions. This includes the IKOS construction (Section 2.2), packed secret sharing (Section 2.3), and a protocol for verifying the consistency of secret shares (Section 2.4). In Section 3, we show a protocol that allows parties to verify that a secret that is shared among them is numerically small. In Section 4 and Section 5 we show our protocols for proofs of plaintext knowledge and proofs of correct multiplication, respectively. To maintain a clear presentation, we defer all proofs to Appendix A.

2 Preliminaries

2.1 Notation

The natural security parameter in this work is λ. We let $\mathbb{Z}_q = \{-q/2, \ldots, q/2\}$ and use $a \mod q$ to denote the mapping of a into the interval $(-q/2, q/2]$. We use $[n]$ to denote the set $\{1, \ldots, n\} \subset \mathbb{Z}$.

We use boldface lower-case letters to represent vectors, such as $\mathbf{u} = (u_1, \ldots, u_n) \in \mathbb{Z}_q^n$. Throughout what follows, *vectors will be assumed to be column vectors, unless stated otherwise.* We use subscripts to denote coordinates on a vector, e.g. u_i is the ith coordinate of vector $\mathbf{u}$. This is to differentiate between coordinates of a vector and elements in a sequence. For the latter case, we use superscripts: $m^{(i)}$ is the ith element of sequence $m^{(1)}, \ldots, m^{(k)}$. We will also sometimes use the notation $(u_i)_{i\in[n]}$ to denote the vector $(u_1, \ldots, u_n)$. We use boldface upper-case letters to represent matrices, such as $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. For a matrix $\mathbf{A}$, we use $\mathbf{A}_i$ to represent its ith row vector, $\mathbf{a}^{(i)}$ to represent its ith column vector, and $a_j^{(i)}$ to represent the element in the ith row and jth column. If $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is a matrix and $R \subseteq [n], C \subseteq [m]$ are sets, we write $\mathbf{A}_R$ and $\mathbf{A}^C$ to denote the matrix $\mathbf{A}$ restricted to the rows with indices in R, and the matrix $\mathbf{A}$ restricted to the columns with indices in C, respectively. We let $\mathbf{A}_R^C = (\mathbf{A}_R)^C = (\mathbf{A}^C)_R$. For a vector $\mathbf{x} = (x_1, \ldots, x_n)$ and a scalar a, we let $a\mathbf{x} = (ax_1, \ldots, ax_n)$.

For a distribution χ, we denote $x \leftarrow \chi$ to be the experiment of choosing x according to χ. If S is a set, then we use $x \leftarrow S$ to denote the experiment of choosing x from the uniform distribution on S. For a randomized function f, we write $f(x\,; r)$ to denote the unique output of f on input x with random coins r. Denote $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ as the group of all reals in $[0, 1)$ with addition modulo 1. For $\alpha \in \mathbb{R}^+$, Ψ_α is defined to be the distribution on $\mathbb{T}$ of a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$, reduced modulo 1. For any probability distribution ϕ over $\mathbb{T}$ and integer $q \in \mathbb{Z}^+$, its *discretization* $\bar{\phi}$ is the discrete distribution over $\mathbb{Z}_q$ of the random variable $\lfloor q \cdot X_\phi \rceil \mod q$, where $X_\phi \leftarrow \phi$.

We use lower case π to denote MPC protocols, such as π_f, and use upper case Π to denote zero-knowledge proof protocols, such as Π_R. We use greek letters to represent shares from a secret sharing. For example, $\boldsymbol{\alpha} = (\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(N)})$ denotes the shares $\alpha^{(i)}$ of each of the N share holders.

2.2 Overview of IKOS Construction

Let $R(x, w)$ be a NP-relation. Consider the following N-player functionality f. The public statement x is known to all players $\mathcal{P}_1, \dots, \mathcal{P}_N$. The functionality takes the entire input w from a special player $\mathcal{I}$ called the "input client", and outputs $R(x, w)$ to all N players. Ishai et al. [IKOS07] show how to construct a zero-knowledge proof protocol for NP-relation R from a MPC protocol π_f for the functionality f described above. We give a high-level idea of the construction. The prover runs the MPC protocol π_f "in his head" and commits to the views $V_1, \dots, V_N$ of the N players. The verifier then chooses a subset $T \subset [N]$, and the prover opens his commitments to views $\{V_i\}_{i \in T}$. The verifier accepts iff the commitment openings are successful, the revealed views are consistent, and the output in each view is 1.

We show the formal statement of the result in Theorem 2.4, but first recall the security properties that the underlying MPC protocol will need to satisfy in the construction. The following definitions are taken almost verbatim from [IKOS07].

Definition 2.1 (Correctness)**.** *We say that a protocol π realizes functionality f with* perfect correctness *if for all inputs (x, w), the probability that the output of some player is different from the output of f is 0, where the probability is taken over the random inputs $r_1, \dots, r_N$.*

Definition 2.2 ((Statistical) t-Privacy)**.** *Let $t \in [N]$. We say a protocol π realizes functionality f with* statistical t-privacy *if there exists a PPT simulator $\mathtt{Sim}$ such that for all inputs (x, w) and all sets of corrupted players $T \subset [N]$ with $|T| \leq t$, the joint view $(\textsc{View}(P_i))_{i \in T}$ of players in T is distributed stastistically close to $\mathtt{Sim}(T, x, R_T(x, w))$.*

Definition 2.3 (t-Robustness)**.** *Let $t \in [N]$. We say a protocol π realizes functionality f with* perfect t-robustness *if it is perfectly correct in the presence of a semi-honest adversary, and for any computationally unbounded malicious adversary corrupting $\mathcal{I}$ and a set T of at most t players, for all inputs x, it holds that if there does not exist w such that $f(x, w) = 1$, then the probability that an uncorrupted player $P_i \notin T$ outputs 1 is 0.*

Theorem 2.4 ([IKOS07])**.** *Let f be the N-player functionality with input client $\mathcal{I}$ described above. Suppose that π_f is a protocol that realizes f with perfect t-robustness (in the malicious model) and statistical t-privacy (in the semi-honest model), where $t = \Omega(\lambda)$, and $N = ct$ for some constant $c > 1$. Given π_f and an unconditionally-binding commitment scheme, it is possible to construct a computational honest-verifier zero-knowledge proof protocol $\Pi_{R,\mathcal{I},t}$ for the NP-relation R, with negligible (in λ) soundness error.*

One of the nice properties about the [IKOS07] construction is that we get *broadcast for free* because the Prover can simply send the broadcasted messages directly to the Verifier. Therefore, the communication cost of broadcasting a message is simply the size of the message. We also get *coin-flipping among the players for free* because the (honest) Verifier can simply provide the random value. Therefore, the communication cost of coin-flipping for a value is simply the size of the value. We will use these two facts in our constructions. Also, as observed by [BD10], if we use a commitment scheme that allows us to commit to strings with only a constant additive length increase such as those implicit in [PVW08], then the zero-knowledge proof protocol $\Pi_{R,\mathcal{I},t}$ (asymptotically) conserves the communication complexity of the underlying MPC protocol π_f.

Finally, using general zero-knowledge techniques, it is possible to convert the honest-verifier zero-knowledge proof protocol $\Pi_{R,\mathcal{I},t}$ obtained from Theorem 2.4 into a full zero-knowledge protocol, while (asymptotically) preserving the communication complexity of the protocol. One such technique is described in [IKOS07].

2.3 Packed Secret Sharing

We will use the packed secret sharing technique of Franklin and Yung [FY92]. Similar to Shamir secret sharing over $\mathbb{Z}_q$ [Sha79], packed secret sharing allows a dealer to share a vector of k values $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ using a single random polynomial of degree at most d. To guarantee security against at most t corrupted players, we must have $d \geq t + k - 1$. The idea is to chose a random polynomial $P(\cdot)$ of degree at most d, subject to the condition $P(-j+1) = x_j$ for $j \in [k]$. The share of player i is, as usual, the value $\alpha_i = P(i)$.

We use $[\mathbf{x}]_d$ to denote a packed secret-sharing $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N) \in \mathbb{Z}_q^N$ for N players of the block $\mathbf{x}$ using a polynomial of degree at most d. We call $[\mathbf{x}]_d$ a *d-sharing* of $\mathbf{x}$. We say $\mathbf{x}$ is *correctly shared* if every honest player $\mathcal{P}_i$ is holding a share α_i of $\mathbf{x}$, such that there exists a degree at most d polynomial $P(\cdot)$ with $P(i) = \alpha_i$ for $i \in N$, and $P(-j+1) = x_j$ for $j \in [k]$. Any (perhaps incomplete) set of shares is called *d-consistent* if these shares lie on a polynomial of degree at most d.

Let $\mathbf{Z} \in \mathbb{Z}_q^{m \times k}$ be a matrix of secrets. Suppose we have d-sharings of the rows of $\mathbf{Z}$: $[\mathbf{Z}_1]_d, \ldots, [\mathbf{Z}_m]_d \in \mathbb{Z}_q^{1 \times N}$. We define $\boldsymbol{\Psi} \in \mathbb{Z}_q^{m \times N}$, called a *$d$-share matrix* of $\mathbf{Z}$, to be a matrix

$$\boldsymbol{\Psi} = \begin{bmatrix} [\mathbf{Z}_1]_d \\ \vdots \\ [\mathbf{Z}_m]_d \end{bmatrix} \in \mathbb{Z}_q^{m \times N}$$

Note that the shares held by $\mathcal{P}_i$ are precisely the entries in the ith column vector of $\boldsymbol{\Psi}$, denoted by $\boldsymbol{\psi}^{(i)}$.

For any function $f : \mathbb{Z}_q^{m \times 1} \to \mathbb{Z}_q^{m' \times 1}$, we abuse notation and write

$$f(\boldsymbol{\Psi}) = f\begin{pmatrix} [\mathbf{Z}_1]_d \\ \vdots \\ [\mathbf{Z}_m]_d \end{pmatrix} = \begin{bmatrix} [\mathbf{Y}_1]_{d'} \\ \vdots \\ [\mathbf{Y}_{m'}]_{d'} \end{bmatrix},$$

to signify that each player $\mathcal{P}_i$ locally applies f to his shares of all $[\mathbf{Z}_j]_d$'s to obtain his share of each $[\mathbf{Y}_j]_{d'}$. In other words, if $\boldsymbol{\Psi}$ is the d-share matrix of $\mathbf{Z}$ then each player locally computes $f(\boldsymbol{\psi}^{(i)}) = \boldsymbol{\phi}^{(i)}$, where $\Phi = [\boldsymbol{\phi}^{(1)}, \ldots, \boldsymbol{\phi}^{(N)}] \in \mathbb{Z}_q^{m' \times N}$ is the d'-share matrix of $\mathbf{Y}$ containing the $\mathbf{Y}_j$'s as rows.

It is easy to see that if $f(\mathbf{x})$ is a linear function and we define f_i to be f with its output restricted to the ith coordinate (i.e. $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_{m'}(\mathbf{x}))^\top$), then

$$f\begin{pmatrix} [\mathbf{Z}_1]_d \\ \vdots \\ [\mathbf{Z}_m]_d \end{pmatrix} = \begin{bmatrix} \left[f_1(\mathbf{z}^{(1)}) \ , \ \ldots \ , \ f_1(\mathbf{z}^{(k)})\right]_d \\ \vdots \\ \left[f_{m'}(\mathbf{z}^{(1)}) \ , \ \ldots \ , \ f_{m'}(\mathbf{z}^{(k)})\right]_d \end{bmatrix}$$

Note that if f is a linear function, then the sharings obtained as a result of applying f are also d-sharings. In particular, if each player $\mathcal{P}_i$ multiplies his share vector $\boldsymbol{\psi}^{(i)}$ by a matrix $\mathbf{M} \in \mathbb{Z}_q^{m' \times m}$, the player obtains a $(m' \times 1)$-vector representing his corresponding shares of:

$$\mathbf{M}\boldsymbol{\Psi} = \begin{bmatrix} \left[\mathbf{M}_1\mathbf{z}^{(1)} \ , \ \ldots \ , \ \mathbf{M}_1\mathbf{z}^{(k)}\right]_d \\ \vdots \\ \left[\mathbf{M}_{m'}\mathbf{z}^{(1)} \ , \ \ldots \ , \ \mathbf{M}_{m'}\mathbf{z}^{(k)}\right]_d \end{bmatrix} = \begin{bmatrix} \left[\left(\mathbf{M}_1\mathbf{z}^{(j)}\right)_{j\in[k]}\right]_d \\ \vdots \\ \left[\left(\mathbf{M}_{m'}\mathbf{z}^{(j)}\right)_{j\in[k]}\right]_d \end{bmatrix} = \begin{bmatrix} [(\mathbf{MZ})_1]_d \\ \vdots \\ [(\mathbf{MZ})_{m'}]_d \end{bmatrix},$$

where $(\mathbf{MZ})_i$ is the ith row of the matrix $\mathbf{MZ}$.

Parameters. We discuss requirements on the parameters of the scheme. We let $N = c_1 t$ for $c_1 > 2$, satisfying the requirements of the IKOS construction. In order to guarantee privacy of the secret shares, we must have $d \geq t + k - 1$. We will sometimes use $(d/2)$-shares, so we assume $d/2 \geq t + k - 1$. Furthermore, we must have enough honest players so that their shares alone can determine a polynomial of degree d (in case corrupt players do not send their shares for reconstruction). We therefore need $N - t \geq d \geq d/2 \geq t + k - 1$. For our choice of N this yields $k \leq (c_1 - 2)t + 1$. Thus, we assume $k = \Theta(t)$. Also, in order to have enough evaluation points, we must have $q > k + N$. Henceforth, we will use this choice of parameters.

2.4 Verifying Consistency of Shares

We now describe a protocol that can be used by N parties to check that their shares are d-consistent. Security is guaranteed if at most $t < N/2$ parties are corrupted. Players check $N - 2t$ sets of shares at a time. More formally, let $\mathbf{Z} \in \mathbb{Z}_q^{(N-2t)\times k}$ be a matrix of secrets, and suppose d-shares $[\mathbf{Z}_1]_d, \ldots [\mathbf{Z}_{N-2t}]_d$ of the rows of $\mathbf{Z}$ are distributed among the N players. The players want to verify that each sharing is d-consistent without revealing their individual shares. Beerliová-Trubíniová and Hirt [BH08] describe a protocol in which the N parties can perform this check when they hold N sharings (as opposed to $N - 2t$, as described here) and sharing $[\mathbf{Z}_i]_d$ was created by player $\mathcal{P}_i$. Bendlin and Damgård [BD10] extend this protocol to the case when all the shares were prepared by a (possibly corrupt) input client $\mathcal{I}$. We describe the protocol of [BD10] in Figure 1. In the protocol, all players receive as common input a *hyper-invertible* matrix $\mathbf{M} \in \mathbb{Z}_q^{N\times(N-t)}$ for $q > 2N$. Informally, a hyper-invertible matrix is a matrix such that every square submatrix of $\mathbf{M}$ is invertible. Beerliová-Trubíniová and Hirt [BH08] show how such matrices can be constructed.

Lemma 2.5. *The protocol π_{CHECK} described in Figure 1 allows N players, at most t of which are corrupted, to verify with zero error probability that $(N - 2t)$ pack-sharings, each of $k = \Theta(t)$ secrets in $\mathbb{Z}_q$, are d-consistent (for $d \geq t + k - 1$). It is t-private in the presence of a semi-honest advesary, t-robust in the presence of a malicious adversary, and has communication complexity $N(N+t)\log q$.*

Protocol $\pi_{\textbf{Check}}$ between parties $(\mathcal{P}_1, \dots, \mathcal{P}_N)$ to verify d-consistency of shares.

Common input: hyper-invertible matrix $\mathbf{M} \in \mathbb{Z}_q^{N \times (N-t)}$
Input to $\mathcal{P}_i$: corresponding shares of $[\mathbf{Z}_1]_d, \dots, [\mathbf{Z}_{(N-2t)}]_d$.

1. Input client $\mathcal{I}$ chooses and d-shares random vectors in $\mathbb{Z}_q^{1 \times k}$. Let $[\mathbf{Z}_{N-2t+1}]_d, \dots [\mathbf{Z}_{N-t}]_d$ be the resulting shares. Augment matrix $\mathbf{Z}$ with rows $\mathbf{Z}_{N-2t+1}, \dots, \mathbf{Z}_{N-t}$ to obtain matrix $\mathbf{Z}' \in \mathbb{Z}_q^{(N-t) \times k}$. Let $\mathbf{\Psi} \in \mathbb{Z}_q^{(N-t) \times N}$ be the d-share matrix of $\mathbf{Z}'$.
2. Players locally compute:
$$\mathbf{\Phi} = \mathbf{M}\mathbf{\Psi} = \begin{bmatrix} [(\mathbf{M}\mathbf{Z}')_1]_d \\ \vdots \\ [(\mathbf{M}\mathbf{Z}')_N]_d \end{bmatrix} \in \mathbb{Z}_q^{N \times N}$$
3. The players reconstruct the resulting shares, each towards a different player: player $\mathcal{P}_i$ receives $\mathbf{\Phi}_i$. Each player verifies that the shares he receives are d-consistent and broadcasts "ABORT" if he finds a fault, and otherwise broadcasts "OK".
4. If all players broadcast "OK" then the players conclude that the initial shares were d-consistent.

Figure 1: Protocol π_{CHECK} to verify consistency of shares

2.5 Regev Encryption Scheme

Before presenting the Regev encryption scheme [Reg05], we first introduce the hardness assumption on which its security is based. For positive integers $n = n(\lambda)$ and $q = q(\lambda) \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}_q$, let $A_{\mathbf{s},\chi}$ be the distribution obtained by choosing $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and $x \leftarrow \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + x) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Learning with Errors (LWE$_{n,q,\chi}$ and dLWE$_{n,q,\chi}$). The Learning with Errors problem LWE$_{n,q,\chi}$ is defined as follows. Given $m = poly(n)$ samples chosen according to $A_{\mathbf{s},\chi}$ for *uniformly chosen* $\mathbf{s} \in \mathbb{Z}_q^n$, output $\mathbf{s}$ with noticeable probability. The Decisional Learning with Errors problem dLWE$_{n,q,\chi}$ is to distinguish (with non-negligible advantage) $m = poly(n)$ samples chosen according to $A_{\mathbf{s},\chi}$ for *uniformly chosen* $\mathbf{s} \in \mathbb{Z}_q^n$, from m samples chosen uniformly at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$. In other words, if dLWE$_{n,q,\chi}$ is hard then $A_{\mathbf{s},\chi}$ is pseudorandom. We will use $\chi = \bar{\Psi}_\alpha$ and in this case, we write LWE$_{n,q,\alpha}$ to mean LWE$_{n,q,\bar{\Psi}_\alpha}$.

Discrete Gaussian Distribution. We present an elementary fact that shows that the discrete Gaussian distribution with standard deviation r outputs an element x with with $||x|| \leq r\sqrt{n}$ with high probability.

Lemma 2.6 (see [MR07], Theorem 4.4)**.** *Let $n \in \mathbb{N}$. For any real number $r > \omega(\sqrt{\log n})$, we have* $\Pr_{\mathbf{x} \leftarrow D_{\mathbb{Z}^n,r}}[||\mathbf{x}|| > r\sqrt{n}] \leq 2^{-n+1}$.

Using Lemma 2.6 together with the fact that for all $\mathbf{x} \in \mathbb{R}^n$, $||\mathbf{x}||_\infty \geq ||\mathbf{x}||/\sqrt{n}$ we arrive at the following bound.

Lemma 2.7. *Let $n \in \mathbb{N}$. For any real number $r > \omega(\sqrt{\log n})$, we have* $\Pr_{\mathbf{x} \leftarrow D_{\mathbb{Z}^n,r}}[||\mathbf{x}||_\infty > r] \leq 2^{-n+1}$.

This allows us to define a *truncated* Gaussian distribution that always outputs (with probability 1) elements with ℓ_∞ norm less than r. Simply define the truncated Gaussian $\overline{D}_{\mathbb{Z}^n,r}$ over $\mathbb{Z}^n$ with standard deviation r to sample a vector according to the discrete Gaussian $D_{\mathbb{Z}^n,r}$ and repeat the sampling if the vector has ℓ_∞ norm greater than r. We will use the truncated discrete Gaussian in our schemes to ensure that samples are bounded by r in each coordinate (and can thus ensure perfect correctness), but state security in terms of the discrete Gaussian. Since the distributions are statistically close, all results stated using the discrete Gaussian also hold when using the truncated distribution.

We present a generalized version of the Regev encryption scheme [Reg05] (with the modifications of [GPV08]), using the truncated discrete Gaussian (as above). The scheme is parametrized by integers $n = n(\lambda), m = m(\lambda) > n, q = q(\lambda), r = r(\lambda)$, and $p = p(\lambda) < q$. The message space is $\mathcal{M} = \mathbb{Z}_p$, the ciphertext space is $\mathcal{C} = (\mathbb{Z}_q^n, \mathbb{Z}_q)$. All operations are performed over $\mathbb{Z}_q$.

- `KeyGen`(1^n): Output $sk = \mathbf{s}, pk = (\mathbf{A}, \mathbf{b})$, where $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{x} \leftarrow \chi^m$, $\mathbf{b} = \mathbf{A}^\top \mathbf{s} + \mathbf{x} \in \mathbb{Z}_q^m$.
- `Enc`$_{pk}(m)$: Output $(\mathbf{u}, c)$, where $\mathbf{r} \leftarrow \overline{D}_{\mathbb{Z}^m,r}$, $\mathbf{u} = \mathbf{A}\mathbf{r} \in \mathbb{Z}_q^{n\times 1}$, $c = \mathbf{b}^\top \mathbf{r} + m \cdot \lfloor q/p \rceil \in \mathbb{Z}_q$.
- `Dec`$_{sk}(\mathbf{u}, \mathbf{c})$: Output $m = \lfloor (c - \mathbf{s}^\top \mathbf{u}) \cdot p/q \rceil$.

Theorem 2.8 ([Reg05, GPV08]). *Let $q \geq 5prm, \alpha \leq 1/(p \cdot r\sqrt{m} \cdot \omega(\sqrt{\log \lambda})), \chi = \bar{\Psi}_\alpha, m \geq 2(n+1)\log q + \omega(\log \lambda)$. With this choice of parameters, the Regev encryption scheme is correct and IND-CPA-secure, assuming LWE$_{n,q,\chi}$ is hard.*

Parameters and Worst-case Guarantees. Our construction requires the modulus q to be super-polynomial in the security parameter λ. More specifically, we require $\sqrt{q/8} > 2^{\omega(\log \lambda)} \cdot m \cdot \max(p/2, r)$. We can use any choice of parameters that satisfies this constraint and keeps the cryptosystem secure.

One option is to let the dimension of the lattice be our security parameter, ie. $n = \lambda$ and set our modulus q to be exponential in the lattice dimension n. Peikert [Pei09] showed that for such a large q, LWE$_{n,q,\alpha}$ is as hard as $\mathsf{GapSVP}_{\tilde{O}(n/\alpha)}$ if q is a product of primes, each of polynomial size. The works of [BD10, BDOZ11] use this choice of parameters.

Another possible choice is to let $n = \lambda^{1/\epsilon}$ for some $\epsilon \in (0,1)$ (e.g. $n = \lambda^2$), $p, r, m = \text{poly}(\lambda)$ and let $q = 2^{n^\epsilon}$ be subexponential in the lattice dimension n. In this case, we can rely on Regev's quantum reduction [Reg05] to $\mathsf{GapSVP}_{\tilde{O}(n/\alpha)}$ or Peikert's classical reduction [Pei09] to $\mathsf{GapSVP}_{\zeta,\gamma}$ where $\gamma(n) \geq n/(\alpha\sqrt{\log n})$, $\zeta(n) \geq \gamma(n)$ and $q \geq \zeta \cdot \omega(\sqrt{\log n/n})$. The work of [AJW11, AJLA+12] uses this choice of parameters.

3 Verifying that Secrets are Numerically Small

At the heart of our constructions of proofs of plaintext knowledge and correct multiplication, we will use a protocol that allows a dealer (in our case the input client $\mathcal{I}$) to prove to the players that the secret that he secret-shared among them is bounded by some publicly known bound B. Formally, let $\mathbf{R} \in \mathbb{Z}_q^{m\times k}$ be a matrix of secrets. And suppose that a dealer has distributed d-sharings of the rows of $\mathbf{R}$: $[\mathbf{R}_1]_d, \ldots, [\mathbf{R}_m]_d$ between N players. We show a protocol π_{VerSM} that allows the dealer to prove to each player $\mathcal{P}_i$, without revealing $\mathbf{R}$, that all secrets in $\mathbf{R}$ are smaller than $B \ll q/2$.

We first have the dealer compute and distribute a sharing $[\mathbf{b}]_d$ of $\mathbf{b} = (B, \ldots, B) \in \mathbb{Z}_q^k$. Players can then compute

$$\begin{bmatrix} [\mathbf{b}]_d \\ \vdots \\ [\mathbf{b}]_d \end{bmatrix} - \begin{bmatrix} [\mathbf{R}_1]_d \\ \vdots \\ [\mathbf{R}_m]_d \end{bmatrix} = \begin{bmatrix} [\mathbf{b} - \mathbf{R}_1]_d \\ \vdots \\ [\mathbf{b} - \mathbf{R}_m]_d \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} [\mathbf{b}]_d \\ \vdots \\ [\mathbf{b}]_d \end{bmatrix} + \begin{bmatrix} [\mathbf{R}_1]_d \\ \vdots \\ [\mathbf{R}_m]_d \end{bmatrix} = \begin{bmatrix} [\mathbf{b} + \mathbf{R}_1]_d \\ \vdots \\ [\mathbf{b} + \mathbf{R}_m]_d \end{bmatrix}$$

Proving that each secret is bounded by B (and thus lies between $-B$ and B) reduces to proving that all the secrets that are pack-shared by each $[\mathbf{b} - \mathbf{R}_i]_d$ and $[\mathbf{b} + \mathbf{R}_i]_d$ for $i \in [m]$, are positive. We thus show a subroutine, described in Figure 3 that allows a dealer to prove that secrets that are pack-shared among players are positive. To do this, we follow an idea of Boudot [Bou00] and use Lagrange's Four-Square Theorem, which states that *every positive number can be written as the sum of four squares* (see e.g. [FR06]). Moreover, these four squares can be efficiently computed [RS86, Lip03]. Suppose the dealer has pack-shared a secret vector $\mathbf{z} \in \mathbb{Z}_q^{1\times k}$. For each coordinate z_j for $j \in [k]$, the dealer finds the four numbers u_j, v_j, w_j, y_j such that $z_j = u_j^2 + v_j^2 + w_j^2 + y_j^2$. We let $\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{w}}, \tilde{\mathbf{y}}$ be the vectors with u_j, v_j, w_j, y_j as the jth coordinate, respectively. The dealer $(d/2)$-shares each of these vectors $[\tilde{\mathbf{u}}]_{d/2}, [\tilde{\mathbf{v}}]_{d/2}, [\tilde{\mathbf{w}}]_{d/2}, [\tilde{\mathbf{y}}_i]_{d/2}$. Similarly, we let $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{y}$ be the vectors with $u_j^2, v_j^2, w_j^2, y_j^2$ as the jth coordinate, respectively. Players can locally compute sharings $[\mathbf{u}]_d, [\mathbf{v}]_d, [\mathbf{w}]_d, [\mathbf{y}]_d$ by squaring their corresponding shares of $[\tilde{\mathbf{u}}]_{d/2}, [\tilde{\mathbf{v}}]_{d/2}, [\tilde{\mathbf{w}}]_{d/2}, [\tilde{\mathbf{y}}_i]_{d/2}$. Each player then computes,

$$[\mathbf{z}]_d - [\mathbf{u}]_d - [\mathbf{v}]_d - [\mathbf{w}]_d - [\mathbf{y}]_d = [\mathbf{z} - \mathbf{u} - \mathbf{v} - \mathbf{w} - \mathbf{y}]_d = [\mathbf{0}]_d$$

and together they check that the result is indeed a pack-sharing of the vector $\mathbf{0} \in \mathbb{Z}^k$.

However, suppose that a cheating dealer chooses $|u_j| > \sqrt{q/2}$. Then $|u_j^2| > q$ and we have wrap-around modulo q, which means that the cheating dealer could convince the players that a secret z_j is positive, without this being true. To ensure this does not happen, we have the dealer prove that each of u_j, v_j, w_j, y_j is bounded by some bound B', which although larger than B, is certainly much smaller than $\sqrt{q/2}$ (in fact, we will need $B' < \sqrt{q/8}$ so that we don't have overflow when adding the four squares).

Our protocol for verifying that numbers are bounded by B' uses techniques from Cramer and Damgård [CD09]. Players check τ shares at a time, where τ should be thought of as the "local security parameter" for the protocol π_{VerBnd}. The players compute a linear combination of their shares (with some noise added) and reconstruct the result, such that if the secrets resulting from this reconstruction are "not too big" then the original secrets (i.e. the entries in $\mathbf{R}$) are also small. To ensure that the reconstructed result does not reveal $\mathbf{R}$, we let the added noise be in an interval that is a factor of 2^τ larger than the entries in $\mathbf{R}$. To guarantee that π_{VerBnd} has statistical (in λ) t-privacy, we set $\tau = \omega(\log \lambda)$. The final bound that we are able to prove is $B' = 2^{2\tau+1}mB$. We will thus need to ensure that $\sqrt{q/8} > 2^{2\tau+1}mB$.

We give full descriptions of the protocol π_{VerSm} in Figure 2, of the subroutine to verify that secrets are positive in Figure 3, and the subroutine to verify that numbers are bounded by B' in Figure 4.

We set $N = \Theta(t)$ as is required for the IKOS construction and for privacy (see Section 2.3), and analyze the communication complexity of the π_{VerSm} protocol. Each share has size at most $\log q$. Each execution of π_{VerBnd} has communication cost $O(\tau N \log q)$: sharing the $\mathbf{X}_i$'s has communication cost $(2\tau - 1)N \log q$, the coin-flipping of $\mathbf{e}$ has communication cost τ since we'll use

Protocol π_{VerSm} between parties $(\mathcal{P}_1, \dots, \mathcal{P}_N)$ and input client $\mathcal{I}$.

Common input: bound B
Input to $\mathcal{I}$: $\mathbf{R} \in \mathbb{Z}_q^{m\times k}$.
Input to $\mathcal{P}_i$: Corresponding shares of $[\mathbf{R}_1]_d, \dots, [\mathbf{R}_m]_d$.

1. $\mathcal{I}$ prepares a d-sharing of $\mathbf{b} = (B, \dots, B) \in \mathbb{Z}_q^k$): $[\mathbf{b}]_d$. $\mathcal{I}$ gives each player its corresponding shares.
2. Players run the subroutine π_{VerPos} (see Figure 3) with

$$\begin{bmatrix} [\mathbf{b}]_d \\ \vdots \\ [\mathbf{b}]_d \end{bmatrix} - \begin{bmatrix} [\mathbf{R}_1]_d \\ \vdots \\ [\mathbf{R}_m]_d \end{bmatrix} = \begin{bmatrix} [\mathbf{b}-\mathbf{R}_1]_d \\ \vdots \\ [\mathbf{b}-\mathbf{R}_m]_d \end{bmatrix} \text{ and } \begin{bmatrix} [\mathbf{b}]_d \\ \vdots \\ [\mathbf{b}]_d \end{bmatrix} + \begin{bmatrix} [\mathbf{R}_1]_d \\ \vdots \\ [\mathbf{R}_m]_d \end{bmatrix} = \begin{bmatrix} [\mathbf{b}+\mathbf{R}_1]_d \\ \vdots \\ [\mathbf{b}+\mathbf{R}_m]_d \end{bmatrix}$$

Figure 2: Protocol π_{VerSm} to verify that secrets are numerically small

Subroutine π_{VerPos} between parties $(\mathcal{P}_1, \dots, \mathcal{P}_N)$ and input client $\mathcal{I}$, to verify that secrets are positive.

Input to $\mathcal{I}$: $\mathbf{Z} \in \mathbb{Z}_q^{m\times k}$.
Input to $\mathcal{P}_i$: Corresponding shares of $[\mathbf{Z}_1]_d, \dots, [\mathbf{Z}_m]_d$.

1. For each entry $z_i^{(j)}$ of $\mathbf{Z}$ (for $i \in [m], j \in [k]$), the dealer finds the four numbers $u_{ij}, v_{ij}, w_{ij}, y_{ij}$ such that $z_i^{(j)} = u_{ij}^2 + v_{ij}^2 + w_{ij}^2 + y_{ij}^2$. Define $\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}, \widetilde{\mathbf{W}}, \widetilde{\mathbf{Y}}$ to be the matrices with $u_{ij}, v_{ij}, w_{ij}, y_{ij}$ as the (i,j)th entry, respectively. Similarly, define $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{Y}$ to be the matrices with $u_{ij}^2, v_{ij}^2, w_{ij}^2, y_{ij}^2$ as the (i,j)th entry, respectively.
2. $\mathcal{I}$ computes and distributes $(d/2)$-sharings of the rows of $\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}, \widetilde{\mathbf{W}}, \widetilde{\mathbf{Y}}$: $[\widetilde{\mathbf{U}}_i]_{d/2}, [\widetilde{\mathbf{V}}_i]_{d/2}, [\widetilde{\mathbf{W}}_i]_{d/2}, [\widetilde{\mathbf{Y}}_i]_{d/2}$, for $i \in [m]$.
3. Players run protocol π_{Check} from Section 2.4 with the shares $[\widetilde{\mathbf{U}}_i]_{d/2}, [\widetilde{\mathbf{V}}_i]_{d/2}, [\widetilde{\mathbf{W}}_i]_{d/2}, [\widetilde{\mathbf{Y}}_i]_{d/2}$, for $i \in [m]$ (a total of $4m/(N-t)$ times) to verify that these shares are $d/2$-consistent.
4. $\mathcal{I}$ and the players run the subroutine π_{VerBnd} (see Figure 4) with the shares $[\widetilde{\mathbf{U}}_i]_{d/2}, [\widetilde{\mathbf{V}}_i]_{d/2}, [\widetilde{\mathbf{W}}_i]_{d/2}, [\widetilde{\mathbf{Y}}_i]_{d/2}$, for $i \in [m]$ (a total of $4m/\tau$ times), to verify that each of the $u_{ij}, v_{ij}, w_{ij}, y_{ij}$ is bounded by $B' < \sqrt{q/8}$.
5. For each row $i \in [m]$, players locally compute d-sharings $[\mathbf{U}_i]_d, [\mathbf{V}_i]_d, [\mathbf{W}_i]_d, [\mathbf{Y}_i]_d$ by squaring their corresponding shares of $[\widetilde{\mathbf{U}}_i]_{d/2}, [\widetilde{\mathbf{V}}_i]_{d/2}, [\widetilde{\mathbf{W}}_i]_{d/2}, [\widetilde{\mathbf{Y}}_i]_{d/2}$.
6. For each row $i \in [m]$, players locally compute

$$[\mathbf{Z}_i]_d - [\mathbf{U}_i]_d - [\mathbf{V}_i]_d - [\mathbf{W}_i]_d - [\mathbf{Y}_i]_d = [\mathbf{Z}_i - \mathbf{U}_i - \mathbf{V}_i - \mathbf{W}_i - \mathbf{Y}_i]_d$$

and check that the result is a pack-sharing of the vector $\mathbf{0} \in \mathbb{Z}^{1\times k}$.

Figure 3: Subroutine π_{VerPos} to verify that secrets are positive

this MPC protocol inside the IKOS construction, and reconstructing $\mathbf{M_e Z}' + \mathbf{X}$ has communication cost $(2\tau - 1)N \log q$. The subroutine π_{VerPos} (Figure 3) has communication complexity $O(mN \log q)$: sharing of the rows of $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{Y}$ has cost $4mN \log q$, the total cost of running π_{Check} is $(N(N+t)\log q) \cdot 4m/(N-2t) = O(mN \log q)$, the total cost of running π_{VerBnd} is $O(\tau N \log q) \cdot 4m/\tau = O(mN \log q)$, and the final reconstruction has cost $mN \log q$. Finally, the communication complexity of protocol π_{VerSm} is $O(mN \log q)$: sharing $\mathbf{b}$ has communication cost

Subroutine π_{VerBnd} between parties $(\mathcal{P}_1,\ldots,\mathcal{P}_N)$ and input client $\mathcal{I}$, to verify that numbers are bounded by $B' = 2^{2\tau+1}mB$.

Common input: bound B
Input to $\mathcal{I}$: $\mathbf{Z}' \in \mathbb{Z}_q^{\tau\times k}$.
Input to $\mathcal{P}_i$: Corresponding shares of $[\mathbf{Z}'_1]_d,\ldots,[\mathbf{Z}'_\tau]_d$ (that are known to be d-consistent).

1. $\mathcal{I}$ chooses $\mathbf{X} \leftarrow [-2^\tau mB, 2^\tau mB]^{(2\tau-1)\times k}$, and prepares d-sharings of the rows of $\mathbf{X}$: $[\mathbf{X}_1]_d,\ldots,[\mathbf{X}_{2\tau-1}]_d$. $\mathcal{I}$ gives each player its corresponding shares.
2. Players $\mathcal{P}_1,\ldots,\mathcal{P}_N$ coin-flip for a random vector $\mathbf{e} \in \{0,1\}^{\tau\times 1}$.
3. Define matrix $\mathbf{M}_\mathbf{e}$ to be the $(2\tau-1)\times\tau$ matrix with its (i,j)-th entry defined by $m^{(j)}_{\mathbf{e},i} = e_{i-j+1}$ for $1 \le i-j+1 \le \lambda$. Each player locally computes
$$\begin{bmatrix} [(\mathbf{M}_\mathbf{e}\mathbf{Z}')_1]_{d'} \\ \vdots \\ [(\mathbf{M}_\mathbf{e}\mathbf{Z}')_{2\tau-1}]_{d'} \end{bmatrix} + \begin{bmatrix} [\mathbf{X}_1]_{d'} \\ \vdots \\ [\mathbf{X}_{2\tau-1}]_{d'} \end{bmatrix} = \begin{bmatrix} [(\mathbf{M}_\mathbf{e}\mathbf{Z}'+\mathbf{X})_1]_{d'} \\ \vdots \\ [(\mathbf{M}_\mathbf{e}\mathbf{Z}'+\mathbf{X})_{2\tau-1}]_{d'} \end{bmatrix}$$
4. Players reconstruct $\mathbf{M}_\mathbf{e}\mathbf{Z}'+\mathbf{X}$ row by row and check that all its entries are bounded by $2^{2\tau+1}mB$.

Figure 4: Subroutine π_{VerBnd} to verify that numbers are bounded by $B' = 2^{2\tau+1}mB < \sqrt{q/8}$

$N\log q$, and we run the subroutine π_{VerPos} twice.

Lemma 3.1. *Let n,m,r,q,N,t,k be as in Theorem 2.8 and Section 2.3, and let B be some publicly-known bound. If $\tau = \omega(\log\lambda)$ and $\sqrt{q/8} > 2^{2\tau+1}mB$ then the protocol π_{VerSM} described in Figure 2 allows N players to verify, with negligible error probability in λ, that all entries in a secret matrix $\mathbf{R} \in \mathbb{Z}_q^{m\times k}$ are bounded by B. It has statistical t-privacy in the presence of a semi-honest adversary, perfect t-robustness in the presence of a malicious adversary, and communication complexity $O(mN\log q)$.*

4 Proofs of Plaintext Knowledge

We wish to show a zero-knowledge proof protocol that allows a prover to prove that he knows the plaintexts of k different ciphertexts, each encrypted under the same public key. We show how to do this for the Regev encryption scheme described in Section 2.5. More formally, we show a zero-knowlege proof protocol for the following relation:

$$\begin{aligned} R_{\text{PoPK}} = \{\ (x,w)\ \mid\ & x = ((\mathbf{A},\mathbf{b}),(\mathbf{u}^{(1)},c^{(1)}),\ldots,(\mathbf{u}^{(k)},c^{(k)})), \\ & w = ((m^{(1)},\mathbf{r}^{(1)}),\ldots,(m^{(k)},\mathbf{r}^{(k)})) \quad \text{s.t.} \\ & \forall\ j \in [k] : (\mathbf{u}^{(j)},c^{(j)}) = \mathtt{Enc}_{(\mathbf{A},\mathbf{b})}(m^{(j)};\mathbf{r}^{(j)}) \\ & \text{and}\ \ |m^{(j)}| \le p/2\ ,\ ||\mathbf{r}^{(j)}||_\infty < r\ \} \end{aligned}$$

We create protocol Π_{PoPK} for relation R_{PoPK} using the "MPC-in-the-head" technique of [IKOS07] described in Section 2.2. We let f_{PoPK} be the N-party functionality that takes the entire input w from $\mathcal{I}$ and outputs $R_{\text{PoPK}}(x,w)$ to all N players. In Figure 5, we show our construction of a t-robust and t-private N-party protocol, π_{PoPK}, realizing functionality f_{PoPK}. The idea is to have $\mathcal{I}$ pack secret-share the messages, as well as pack secret-share each coordinate of the randomness vectors. The players then locally run the encryption algorithm on their shares, reconstruct the resulting shares, and check that the reconstructed secrets are indeed the claimed ciphertexts. The input client $\mathcal{I}$ also needs to prove that the messages and randomness come from the correct spaces. For example, he would need to show that the magnitude of each message is less than $p/2$ (since the message space is $\mathbb{Z}_p$), and that each coordinate of each randomness vector is at most r (since we are using the truncated Gaussian distribution described in Section 2.5). For this, we will use the protocol π_{VerSm} described in Section 3.

We set $t = \Theta(k)$ and $N = \Theta(t)$ as is required for the IKOS construction and for privacy (see Section 2.3), and analyze the communication complexity of our protocol π_{PoPK} (see Figure 5). Since each share has size $\log q$, step 1 has communication cost $(m+1)N \log q = O(mk \log q)$. We run π_{Check} $m + 1/(N-2t) = O(m/k)$ times, so step 2 has communication cost $N(N+t)\log q(m/k) = O(mk \log q)$ The reconstruction in step 3 has cost $2nN \log q$ and running protocol π_{VerSm} has cost $2mN \log q$ so the total cost of step 3 and of π_{PoPK} is $O(mk \log q)$.

Our techniques are similar to those of Bendlin and Damgård [BD10]. However, our protocol π_{VerSm} for proving that a secret is small (see Section 3) allows us to prove soundness for message space $\mathbb{Z}_p$ and randomness sampled from the discrete Gaussian, whereas the construction of [BD10] only worked for bit messages and bit-vector randomness. Finally, our use of packed secret sharing allows us to achieve a better amortized communication complexity. The protocol of [BD10] has complexity $O(nm \log q)$ per proof, whereas we achieve an amortized complexity of $O(m \log q)$ per proof.

Lemma 4.1. *Let n, m, r, p, q, N, t, k be as in Lemma 3.1 with $B = \max(p/2, r)$. The protocol π_{PoPK} described in Figure 5 realizes f_{PoPK} with statistical t-privacy in the presence of a semi-honest adversary and perfect t-robustness in the presence of a malicious adversary, and has communication complexity $O(mk \log q)$.*

Putting together Lemma 4.1 with Theorem 2.4 yields the following theorem.

Theorem 4.2. *Let n, m, r, p, q be as in Lemma 3.1 with $B = \max(p/2, r)$. Given an unconditionally-binding commitment scheme, it is possible to construct a computational zero-knowledge proof protocol Π_{PoPK} for relation R_{PoPK} with negligible (in λ) soundness error and amortized communication complexity $O(m \log q)$ per proof.*

5 Proofs of Correct Multiplication

In this section we show proofs for correct multiplication for the Regev encryption scheme. In our protocol, the prover performs k proofs at a time, all under the same public key. More formally, we give a zero-knowledge proof protocol for the following relation:

$$\begin{aligned} R_{\text{PoCM}} = \{\ (x, w)\ \mid\ & x = ((\mathbf{A}, \mathbf{b}), (\mathbf{u}^{(1)}, c^{(1)}, \mathbf{v}^{(1)}, e^{(1)}), \ldots, (\mathbf{u}^{(k)}, c^{(k)}, \mathbf{v}^{(k)}, e^{(k)})), \\ & w = ((m^{(1)}, \mathbf{r}^{(1)}, x^{(1)}), \ldots, (m^{(k)}, \mathbf{r}^{(k)}, x^{(k)})) \quad \text{s.t.} \\ & \forall j \in [k]:\ (\mathbf{v}^{(j)}, e^{(j)}) = m^{(j)}(\mathbf{u}^{(j)}, c^{(j)}) + \mathtt{Enc}_{(\mathbf{A},\mathbf{b})}(x^{(j)}; \mathbf{r}^{(j)}) \\ & \text{and}\ \ |m^{(j)}| \le p/2\ ,\ |x^{(j)}| \le p/2\ ,\ ||\mathbf{r}^{(j)}||_\infty < r\ \} \end{aligned}$$

As in Section 4, we create protocol Π_{PoCM} for relation R_{PoCM} using the "MPC-in-the-head" technique of [IKOS07], described in Section 2.2. We let f_{PoCM} be the N-party functionality that

Protocol π_{PoPK} between parties $(\mathcal{P}_1, \dots, \mathcal{P}_N)$ and input client $\mathcal{I}$.

Common input: $p, q, R, x = ((\mathbf{A}, \mathbf{b}), (\mathbf{u}^{(1)}, c^{(1)}), \dots, (\mathbf{u}^{(k)}, c^{(k)}))$
Input to $\mathcal{I}$: $w = ((m^{(1)}, \mathbf{r}^{(1)}), \dots, (m^{(k)}, \mathbf{r}^{(k)}))$

1. Input client $\mathcal{I}$ prepares and distributes among the N players, d-shares over $\mathbb{Z}_q$ of the messages and randomness vectors, with $d = k + t - 1$. The ith coordinates of all randomness vectors are pack-shared to produce a single set of shares $\boldsymbol{\rho}_i$. More formally: define matrices $\mathbf{R} = [\mathbf{r}^{(1)} \; ; \; \mathbf{r}^{(2)} \; ; \; \dots \; ; \; \mathbf{r}^{(k)}] \in \mathbb{Z}_q^{m \times k}$, $\mathbf{m} = [m^{(1)} \; ; \; \dots \; ; \; m^{(k)}] \in \mathbb{Z}_p^{1 \times k}$, $\mathbf{U} = [\mathbf{u}^{(1)} \; ; \; \mathbf{u}^{(2)} \; ; \; \dots \; ; \; \mathbf{u}^{(k)}] \in \mathbb{Z}_q^{n \times k}$, and $\mathbf{c} = (c^{(1)} \; ; \; \dots \; ; \; c^{(k)}) \in \mathbb{Z}_q^{1 \times k}$. $\mathcal{I}$ prepares and distributes d-shares $[\mathbf{m}]_d, [\mathbf{R}_1]_d, \dots, [\mathbf{R}_m]_d$.
2. Players run protocol π_{Check} from Section 2.4 (possibly several times) to verify that their shares are d-consistent.
3. Players "emulate" encryption by running the encryption algorithm on their local shares. More formally:
   - For $\ell \in [n]$, players locally compute $\left[\left(\mathbf{A}_\ell \mathbf{r}^{(j)}\right)_{j \in k}\right]_d$, and check that the result is a pack-sharing of $\mathbf{U}_\ell$.
   - Similarly, players locally compute
   $$\left[\left(\mathbf{b}\mathbf{r}^{(j)}\right)_{j \in k}\right]_d + \left\lfloor \frac{q}{p} \right\rfloor [\mathbf{m}]_d = \left[\left(\mathbf{b}\mathbf{r}^{(j)} + \left\lfloor \frac{q}{p} \right\rfloor m^{(j)}\right)_{j \in k}\right]_d$$
   Players check that the result is a pack-sharing of $\mathbf{c}$.
   - Players use π_{VerSm} from Section 3 to check that $|m^{(j)}| \le p/2$ and $||\mathbf{r}^{(j)}||_\infty < r$ for all $j \in [k]$.

Figure 5: MPC protocol π_{PoPK} that realizes f_{PoPK}

takes the entire input w from $\mathcal{I}$ and outputs $R_{\text{PoCM}}(x, w)$ to all N players. In Figure 6, we show our construction of a t-robust and t-private N-party protocol, π_{PoCM}, realizing functionality f_{PoCM}. Again, the idea is to have $\mathcal{I}$ pack secret-share the messages, as well as pack secret-share each coordinate of the randomness vectors. The players then locally emulate the encryption of the random message and perform the multiplication, then reconstruct the resulting shares, and check that the reconstructed secrets are indeed the claimed ciphertexts. As before, the input client $\mathcal{I}$ also needs to prove that the messages and randomness come from the correct spaces. We again use the protocol π_{VerSm} described in Section 3 for this purpose.

We set $t = \theta(k)$ and $N = \theta(t)$ as is required for the IKOS construction and for privacy (see Section 2.3), and analyze the communication complexity of π_{PoCM} described in Figure 6. Since each share has size $\log q$, step 1 has communication cost $2(m + 1)N \log q = O(mk \log q)$. We run π_{Check} $m + 1/(N - 2t) = O(m/k)$ times, so step 2 has communication cost $N(N + t) \log q(m/k) = O(mk \log q)$. The reconstruction in step 3 has cost $2nN \log q$ and running protocol π_{VerSm} has cost $2mN \log q$ so the total cost of step 3 and of π_{PoPK} is $O(mk \log q)$.

Lemma 5.1. *Let n, m, r, p, q, N, t, k be as in Lemma 3.1 with $B = \max(p/2, r)$. The protocol π_{PoCM} described in Figure 6 realizes f_{PoCM} with statistical t-privacy in the presence of a semi-honest adversary and perfect t-robustness in the presence of a malicious adversary, and has communication complexity $O(mk \log q)$.*

Protocolπ_{PoCM} between parties $(\mathcal{P}_1, \dots, \mathcal{P}_N)$ and input client $\mathcal{I}$.

Common input: $p, q, R, x = ((\mathbf{A}, \mathbf{b}), (\mathbf{u}^{(1)}, c^{(1)}, \mathbf{v}^{(1)}, e^{(1)}), \dots, (\mathbf{u}^{(k)}, c^{(k)}, \mathbf{v}^{(k)}, e^{(k)}))$
Input to $\mathcal{I}$: $w = ((m^{(1)}, \mathbf{r}^{(1)}, x^{(1)}), \dots, (m^{(k)}, \mathbf{r}^{(k)}, x^{(k)}))$

1. Input client $\mathcal{I}$ prepares and distributes among the N players, d-shares over $\mathbb{Z}_q$ of the messages and randomness vectors, with $d = k + t - 1$. The ith coordinates of all randomness vectors are packed shared to produce a single set of shares.. More formally: define matrices $\mathbf{R} = [\mathbf{r}^{(1)} \ ; \ \mathbf{r}^{(2)} \ ; \ \dots \ ; \ \mathbf{r}^{(k)}] \in \mathbb{Z}_q^{m \times k}$, $\mathbf{m} = [m^{(1)} \ ; \ \dots \ ; \ m^{(k)}] \in \mathbb{Z}_p^{1 \times k}, \mathbf{x} = [x^{(1)} \ ; \ \dots \ ; \ x^{(k)}] \in \mathbb{Z}_q^{1 \times k}, \mathbf{U} = [\mathbf{u}^{(1)} \ ; \ \mathbf{u}^{(2)} \ ; \ \dots \ ; \ \mathbf{u}^{(k)}] \in \mathbb{Z}_q^{n \times k}$, $\mathbf{c} = (c^{(1)}, \dots, c^{(k)}) \in \mathbb{Z}_q^{1 \times k}$, $\mathbf{V} = [\mathbf{v}^{(1)} \ ; \ \mathbf{v}^{(2)} \ ; \ \dots \ ; \ \mathbf{v}^{(k)}] \in \mathbb{Z}_q^{n \times k}$, and $\mathbf{e} = (e^{(1)} \ ; \ \dots \ ; \ e^{(k)}) \in \mathbb{Z}_q^{1 \times k}$. $\mathcal{I}$ prepares and distributes $(d/2)$-share $[\mathbf{m}]_d$ and d-shares $[\mathbf{x}]_d, [\mathbf{R}_1]_d, \dots, [\mathbf{R}_m]_d$. $\mathcal{I}$ also prepares and *broadcasts* $(d/2)$-shares$[\mathbf{c}]_{d/2}, [\mathbf{U}_1]_{d/2}, \dots, [\mathbf{U}_n]_{d/2}$.
2. Players run protocol π_{Check} from Section 2.4 (possibly several times) to verify that shares $[\mathbf{x}]_d, [\mathbf{R}_1]_d, \dots, [\mathbf{R}_m]_d$ are d-consistent, and share $[\mathbf{m}]_d$ is $(d/2)$-consistent. They also check locally that $\mathbf{c}, \mathbf{U}_1, \dots, \mathbf{U}_m$ are correctly shared.
3. Players "emulate" correct computation of each $(\mathbf{v}^{(i)}, c^{(i)})$. More formally:
   - For $\ell \in [n]$, players locally compute $\left[\left(\mathbf{A}_\ell \mathbf{r}^{(j)}\right)_{j \in k}\right]_d$. They also locally compute $\left[\left(\mathbf{b}\mathbf{r}^{(j)} + \left\lfloor \frac{q}{p} \right\rfloor x^{(j)}\right)_{j \in k}\right]_d$.
   - For $\ell \in [n]$, players locally compute
   $$[\mathbf{m}]_{d/2} [\mathbf{U}_\ell]_{d/2} + \left[\left(\mathbf{A}_\ell \mathbf{r}^{(j)}\right)_{j \in k}\right]_d = \left[\left(u_\ell^{(j)} m^{(j)} + \mathbf{A}_\ell \mathbf{r}^{(j)}\right)_{j \in k}\right]_d$$
   Players check that the result is a pack-sharing of $\mathbf{V}_\ell$.
   - Players locally compute
   $$[\mathbf{m}]_{d/2} [\mathbf{c}]_{d/2} + \left[\left(\mathbf{b}\mathbf{r}^{(j)} + \left\lfloor \frac{q}{p} \right\rfloor x^{(j)}\right)_{j \in k}\right]_d$$
   $$= \left[\left(c^{(j)} m^{(j)} + \mathbf{b}\mathbf{r}^{(j)} + \left\lfloor \frac{q}{p} \right\rfloor x^{(j)}\right)_{j \in k}\right]_d$$
   Players check that the result is a pack-sharing of $\mathbf{e}$..
   - Players use π_{VerSm} from Section 3 to check that $|m^{(j)}| \le p/2$, $|x^{(j)}| \le p/2$ and $||\mathbf{r}^{(j)}||_\infty < r$ for all $j \in [k]$.

Figure 6: MPC protocol π_{PoCM} that realizes f_{PoCM}

Putting Lemma 5.1 together with Theorem 2.4 yields the following theorem.

Theorem 5.2. *Let n, m, r, p, q be as in Lemma 3.1 with $B = \max(p/2, r)$. Given an unconditionally-binding commitment scheme, it is possible to construct a computational zero-knowledge proof protocol Π_{PoCM} for relation R_{PoCM} with negligible (in λ) soundness error and amortized communication complexity $O(m \log q)$ per proof.*

References

[AJLA$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 483–501. Springer, 2012.

[AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. *Cryptology ePrint Archive: Report 2011/613*, 2011.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC*, volume 5978 of *LNCS*, pages 201–218. Springer, 2010.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *LNCS*, pages 169–188. Springer, 2011.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *LNCS*, pages 420–432. Springer, 1991.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS*, pages 309–325. ACM, 2012.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BH08] Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure mpc with linear communication complexity. In Ran Canetti, editor, *TCC*, volume 4948 of *LNCS*, pages 213–230. Springer, 2008.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *LNCS*, pages 431–444. Springer, 2000.

[BTHN10] Zuzana Beerliová-Trubíniová, Martin Hirt, and Jesper Buus Nielsen. On the theoretical gap between synchronous and asynchronous mpc protocols. In Andréa W. Richa and Rachid Guerraoui, editors, *PODC*, pages 211–218. ACM, 2010.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In Rafail Ostrovsky, editor, *FOCS*, pages 97–106. IEEE, 2011.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[CD09] Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 177–191. Springer, 2009.

[DO10] Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *LNCS*, pages 558–576. Springer, 2010.

[DPSZ11] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:535, 2011.

[FR06] Benjamin Fine and Gerhard Rosenberger. *Number Theory: An Introduction via the Distribution of Primes.* Birkhäuser, 2006.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710. ACM, 1992.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[HM01] Martin Hirt and Ueli M. Maurer. Robustness for free in unconditional multi-party computation. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 101–118. Springer, 2001.

[HNP08] Martin Hirt, Jesper Buus Nielsen, and Bartosz Przydatek. Asynchronous multi-party computation with quadratic communication. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 473–485. Springer, 2008.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 21–30. ACM, 2007.

[KK07] Jonathan Katz and Chiu-Yuen Koo. Round-efficient secure computation in point-to-point networks. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *LNCS*, pages 311–328. Springer, 2007.

[Lip03] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *LNCS*, pages 398–415. Springer, 2003.

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

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *STOC*, pages 333–342. ACM, 2009.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO*, volume 5157 of *LNCS*, pages 554–571. Springer, 2008.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.

[RS86] Michael O. Rabin and Jeffery O. Shallit. Randomized algorithms in number theory. *Communications on Pure and Applied Mathematics*, 39(S1):S239–S259, 1986.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

A Proofs

Proof of Lemma 2.5: We first show $\pi_{\textsc{Check}}$ is t-robust in the presence of a malicious adversary. If all players broadcast "OK", then in particular any $(N-t)$ honest players broadcast "OK", which means that the $(N-t)$ sharings that these players verified are indeed d-consistent. Let $H \subset [N]$ be a set of $(N-t)$ honest players. If players in H broadcast "OK" then we know the rows of Φ_H are d-consistent. By the hyperinvertibility of $\mathbf{M}$, $\mathbf{\Psi}$ can be written as a linear function of $\mathbf{\Phi}_H$ ($\mathbf{\Psi} = \mathbf{M}_H^{-1}\mathbf{\Phi}_H$), which means that if the honest players in H all broadcast "OK" then all the original sharings in $\mathbf{\Psi}$ were d-consistent.

We now argue that $\pi_{\textsc{Check}}$ is t-private in the presence of a semi-honest adversary. Because the adversary is semi-honest, we are guaranteed that the t sharings created in Step 1 are indeed random and d-consistent. Let $C \subset [N]$ be any set of t players, and let $T = \{N-2t+1, \dots, N-t\}$ and $\overline{T} = [N] \backslash T$, so that the rows of $\mathbf{\Psi}_T$ were the random sharings created by $\mathcal{I}$ in Step 1. The hyperinvertibility of $\mathbf{M}$ guarantees that for a fixed set of input shares to be verified (those in $\mathbf{\Psi}_{\overline{T}}$), there is a bijection between $\mathbf{\Phi}_C$ and $\mathbf{\Psi}_T$:

$$\mathbf{\Phi}_C = \mathbf{M}_C\mathbf{\Psi} = \mathbf{M}_C^T\mathbf{\Psi}_T + \mathbf{M}_C^{\overline{T}}\mathbf{\Psi}_{\overline{T}}$$
$$\mathbf{\Psi}_T = (\mathbf{M}_C^T)^{-1}(\mathbf{\Phi}_C - \mathbf{M}_C^{\overline{T}}\mathbf{\Psi}_{\overline{T}})$$

Thus, the joint view of any t players is random and independent of the input shares.

We now discuss the communication complexity of the $\pi_{\textsc{Check}}$ protocol (Figure 1). Each share is of size $\log q$. In Step 1, tN are sent between players, and in Step 3, N^2 shares are sent. Therefore, the communication complexity of $\pi_{\textsc{Check}}$ is $N(N+t)\log q$. □

Proof of Lemma 4.1: We prove that the protocol $\pi_{\textsc{PoPK}}$ shown in Figure 5 is t-private in the presence of a semi-honest adversary and t-robust.

t-Privacy: Let $T \subset [N]$. We describe the simulator $\texttt{Sim}(T, x, R_T(x,w))$. Because we assume a semi-honest adversary, we assume $R(x,w) = 1$, and simply write $\texttt{Sim}(T,x)$. We describe the simulator in Figure 7.

Lemma 2.5 and Lemma 3.1 guarantee that the outputs of the simulators of $\pi_{\textsc{Check}}$ and $\pi_{\textsc{VerSM}}$ are statistically close to the view of the cheating parties in a real execution of the protocol.

Simulator $\mathtt{Sim}(T,x)$ for $\pi_{\mathbf{PoPK}}$

1. Parses $x = ((\mathbf{A},\mathbf{b}),(\mathbf{u}^{(1)},c^{(1)}),\ldots,(\mathbf{u}^{(k)},c^{(k)}))$. Define matrices $\mathbf{U} = [\mathbf{u}^{(1)} \ ; \ \mathbf{u}^{(2)} \ ; \ \ldots \ ; \ \mathbf{u}^{(k)}] \in \mathbb{Z}_q^{n\times k}$, and $\mathbf{c} = (c^{(1)} \ ; \ \ldots \ ; \ c^{(k)}) \in \mathbb{Z}_q^{1\times k}$.
2. Chooses random shares $\mu^{(j)} \leftarrow \mathbb{Z}_q, \boldsymbol{\rho}^{(j)} \leftarrow \mathbb{Z}_q^m$ for all $j \in T$ as the shares for the corrupted parties.
3. Simulates encryption on these shares to obtain "shares" $\boldsymbol{\alpha}^{(j)} \in \mathbb{Z}_q^m, \beta^{(j)} \in \mathbb{Z}_q$ of the ciphertexts:
$$\alpha_\ell^{(j)} = \mathbf{A}_\ell\boldsymbol{\rho}^{(j)} \quad , \quad \beta^{(j)} = \mathbf{b}\rho^{(j)} + \left\lfloor \frac{q}{p} \right\rfloor \mu^{(j)}$$
4. For $\ell \in [m]$, let P_ℓ be a degree $d \geq t+k-1$ polynomial that is consistent with secrets $\mathbf{U}_\ell$ and share $\alpha_\ell^{(j)}$ corresponding to player $\mathcal{P}_j$ for all $j \in T$. Similarly, let P be a degree $d \geq t+k-1$ polynomial that is consistent ith secrets $\mathbf{c}$ and share $\beta^{(j)}$ corresponding to player $\mathcal{P}_j$ for all $j \in T$.
5. For $i \notin T$, let $\alpha_\ell^{(i)} = P_\ell(i)$ and $\beta^{(i)} = P_\ell(i)$.
6. Runs the simulators for π_{Check} and π_{VerSm}.
7. Outputs the output of the simulators for π_{Check} and π_{VerSm}, as well as all the shares $\alpha_\ell^{(j)}, \beta^{(j)}$ for $\ell \in [m], \jmath \in [N]$.

Figure 7: Simulator $\mathtt{Sim}(T,x)$ for π_{PoPK}

Thus, we need only worry about the shares output by the simulator. But by properties of pack secret-sharing, these clearly have the same distribution as those output in a real execution (random subject to lying on a d-degree polynomial corresponding to secret $\mathbf{U}_\ell$ (resp. $\mathbf{c}$)).

t-Robustness: Perfect completeness of π_{Check} and π_{VerSm} (together with the fact that we are using the truncated Gaussian so the randomness of a valid encryption is guaranteed to be bounded by r), as well as the properties of packed secret-sharing described in Section 2.3 guarantee perfect completeness of π_{PoPK}. Now suppose an honest player outputs 1. t-robustness of π_{Check} guarantees that all shares distributed by $\mathcal{I}$ in Step 1 are d-consistent. At reconstruction, players also implicitly verify that all shares in the construction of $\mathbf{U}_\ell$ and $\mathbf{c}$ are d-consistent. Furthermore, recall that we chose $N-t > d$ so that the shares of the honest players uniquely determine a degree-d polynomial, and thus, uniquely determine the shares of the corrupt players. These must be the ones obtained by running the protocol honestly. Thus, there exist $\mathbf{R} = [\mathbf{r}^{(1)} \ ; \ \mathbf{r}^{(2)} \ ; \ \ldots \ ; \ \mathbf{r}^{(k)}] \in \mathbb{Z}_q^{m\times k}$, $\mathbf{m} = [m^{(1)} \ ; \ \ldots \ ; \ m^{(k)}] \in \mathbb{Z}_p^{1\times k}$, namely, those corresponding to the shares received by the players in Step 1, such that $\mathbf{A}_\ell\mathbf{r}^{(j)} = u_\ell^{(j)}$ and $\mathbf{b}\mathbf{r}^{(j)} + \lfloor q/p \rfloor \cdot m^{(j)} = c^{(j)}$ for all $\ell \in [m], j \in [k]$. Furthermore, t-robustness of π_{VerSm} guarantees that $|m^{(j)}| \leq p/2 \quad , \quad ||\mathbf{r}^{(j)}||_\infty < r \quad$ for all $\ j \in [k]$. Thus, if an honest player outputs 1, each encryption in the statement is a valid Regev encryption.

□

Proof of Lemma 5.1: We prove that the protocol π_{PoCM} shown in Figure 6 is t-private in the presence of a semi-honest adversary and t-robust, for $t < N/2$.

t-Privacy: Let $T \subset [N]$. We describe the simulator $\mathtt{Sim}(T,x,R_T(x,w))$. Because we assume a semi-honest adversary, we assume $R(x,w) = 1$, and simply write $\mathtt{Sim}(T,x)$. We describe the simulator in Figure 8.

Simulator $\texttt{Sim}(T, x)$ for $\pi_{\textbf{PoCM}}$

1. Parses $x = ((\mathbf{A}, \mathbf{b}), (\mathbf{u}^{(1)}, c^{(1)}), \dots, (\mathbf{u}^{(k)}, c^{(k)}), (\mathbf{v}^{(1)}, e^{(1)}), \dots, (\mathbf{v}^{(k)}, e^{(k)}))$. Define matrices $\mathbf{U} = [\mathbf{u}^{(1)} \ ; \ \mathbf{u}^{(2)} \ ; \ \dots \ ; \ \mathbf{u}^{(k)}] \in \mathbb{Z}_q^{n \times k}$, and $\mathbf{c} = (c^{(1)} \ ; \ \dots \ ; \ c^{(k)}) \in \mathbb{Z}_q^{1 \times k}$, $\mathbf{V} = [\mathbf{v}^{(1)} \ ; \ \mathbf{v}^{(2)} \ ; \ \dots \ ; \ \mathbf{v}^{(k)}] \in \mathbb{Z}_q^{n \times k}$, and $\mathbf{e} = (e^{(1)} \ ; \ \dots \ ; \ e^{(k)}) \in \mathbb{Z}_q^{1 \times k}$.
2. Chooses random shares $\chi^{(j)}, \mu^{(j)} \leftarrow \mathbb{Z}_q, \boldsymbol{\rho}^{(j)} \leftarrow \mathbb{Z}_q^m$ for all $j \in T$ as the shares for the corrupted parties. Furthermore, honestly creates $d/2$-sharings of $\mathbf{c}$: $\boldsymbol{\gamma} = [\mathbf{c}]_{d/2}$ and the rows of $\mathbf{U}$: $\boldsymbol{\upsilon}_1 = [\mathbf{U}_1]_{d/2}, \dots, \boldsymbol{\upsilon}_m = [\mathbf{U}_m]_{d/2}$.
3. Simulates multiplication on these shares to obtain "shares" $\boldsymbol{\alpha}^{(j)} \in \mathbb{Z}_q^m, \beta^{(j)} \in \mathbb{Z}_q$ of the resulting ciphertext:
$$\alpha_\ell^{(j)} = \mathbf{A}_\ell \boldsymbol{\rho}^{(j)} + \mu^{(j)} \upsilon_\ell^{(j)} \quad , \quad \beta^{(j)} = \mathbf{b}\rho^{(j)} + \left\lfloor \frac{q}{p} \right\rfloor \chi^{(j)} + \mu^{(j)} \gamma^{(j)}$$
4. For $\ell \in [m]$, let P_ℓ be a degree $d \geq t + k - 1$ polynomial that is consistent with secrets $\mathbf{V}_\ell$ and share $\alpha_\ell^{(j)}$ corresponding to player $\mathcal{P}_j$. Similarly, let P be a degree $d \geq t + k - 1$ polynomial that is consistent ith secrets $\mathbf{e}$ and share $\beta^{(j)}$ corresponding to player $\mathcal{P}_j$ for all $j \in T$.
5. For $i \notin T$, let $\alpha_\ell^{(i)} = P_\ell(i)$ and $\beta^{(i)} = P_\ell(i)$.
6. Runs the simulators for π_{Check} and π_{VerSm}.
7. Outputs the output of the simulators for π_{Check} and π_{VerSm}, as well as all the shares $\alpha_\ell^{(j)}, \beta^{(j)}$ for $\ell \in [m], \jmath \in [N]$.

Figure 8: Simulator $\texttt{Sim}(T, x)$ for π_{PoCM}

Lemma 2.5 and Lemma 3.1 guarantee that the outputs of the simulators of π_{Check} and π_{VerSm} are statistically close to the view of the cheating parties in a real execution of the protocol. Thus, we need only worry about the shares output by the simulator. But by properties of pack secret-sharing, these clearly have the same distribution as those output in a real execution (random subject to lying on a d-degree polynomial corresponding to secret $\mathbf{V}_\ell$ (resp. $\mathbf{e}$)).

t-Robustness: Perfect completeness of π_{Check} and π_{VerSm} (together with the fact that we are using the truncated Gaussian so the randomness of a valid encryption is guaranteed to be bounded by r), as well as the properties of packed secret-sharing described in Section 2.3 guarantee perfect completeness of π_{PoCM}. Now suppose an honest player outputs 1. t-robustness of π_{Check} guarantees that all shares distributed by $\mathcal{I}$ in Step 1 are d-consistent. At reconstruction, players also implicitly verify that all shares in the construction of $\mathbf{V}_\ell$ and $\mathbf{e}$ are d-consistent. Furthermore, recall that we chose $N - t > d$ so that the shares of the honest players uniquely determine a degree-d polynomial, and thus, uniquely determine the shares of the corrupt players. These must be the ones obtained by running the protocol honestly. Thus, there exist $\mathbf{x} = [x^{(1)} \ ; \ \dots \ ; \ x^{(k)}] \in \mathbb{Z}_q^{1 \times k}, \mathbf{R} = [\mathbf{r}^{(1)} \ ; \ \mathbf{r}^{(2)} \ ; \ \dots \ ; \ \mathbf{r}^{(k)}] \in \mathbb{Z}_q^{m \times k}$, $\mathbf{m} = [m^{(1)} \ ; \ \dots \ ; \ m^{(k)}] \in \mathbb{Z}_p^{1 \times k}$, namely, those corresponding to the shares received by the players in Step 1, such that $\mathbf{A}_\ell \mathbf{r}^{(j)} + u_\ell^{(j)} m^{(j)} = v_\ell^{(j)}$ and $\mathbf{b}\mathbf{r}^{(j)} + \lfloor q/p \rfloor \cdot x^{(j)} + c^{(j)} m^{(j)} = e^{(j)}$ for all $\ell \in [m], j \in [k]$. Furthermore, t-robustness of π_{VerSm} guarantees that $|x^{(j)}| \leq p/2$, $|m^{(j)}| \leq p/2$, $||\mathbf{r}^{(j)}||_\infty < r$ for all $j \in [k]$. Thus, if an honest player outputs 1, each encryption was the result of a correct multiplication.

□

Proof of Lemma 3.1: To prove Lemma 3.1 we show that the subroutine π_{VerPos} (Figure 3) is statistically t-private in the presence of a semi-honest adversary and t-robust. t-privacy and t-robustness of π_{VerSm} follows directly from this fact. To prove these properties for π_{VerPos}, we first prove them for π_{VerBnd} (Figure 4).

Claim A.1. *The subroutine π_{VerBnd} described in Figure 4 is statistically t-private in the presence of a semi-honest adversary and t-robust, for $t < N/2$.*

Proof. We show statistically t-privacy in the presence of a semi-honest adversary and t-robustness, for $t < N/2$.

t-Privacy: Let $T \subset [N]$. We describe the simulator $\texttt{Sim}(T, x, R_T(x, w))$. Because we assume a semi-honest adversary, we assume $R(x, w) = 1$, and simply write $\texttt{Sim}(T, x)$. We describe the simulator in Figure 9.

Simulator $\texttt{Sim}(T, x)$ for π_{VerBnd}

1. Let $\zeta_1^{(j)}, \dots, \zeta_\tau^{(j)}$ be the input to player $\mathcal{P}_j$.
2. Chooses $\mathbf{e} \leftarrow \{0,1\}^\tau$.
3. Chooses $\mathbf{T} \leftarrow [-2^\tau mB, 2^\tau mB]^{(2\tau-1)\times k}$, and computes $[\mathbf{T}_1]_d, \dots, [\mathbf{T}_{2\tau-1}]_d$. Let $\psi_i^{(j)}$ be the share of $[\mathbf{T}_i]_d$ corresponding to player $\mathcal{P}_j$.
4. For $i \in [\tau]$ and $j \in T$, computes share $\chi_i^{(j)} = \psi_i^{(j)} - \mathbf{M}_\mathbf{e}\zeta_i^{(j)}$ as $\mathcal{P}_j$'s "share" of $\mathbf{X}_i$.
5. Outputs all the shares $[\mathbf{T}_1]_d, \dots, [\mathbf{T}_{2\tau-1}]_d$ and $\chi_i^{(j)}$ for $i \in [\tau]$ and $j \in T$.

Figure 9: Simulator $\texttt{Sim}(T, x)$ for π_{VerBnd}

The distribution of $\mathbf{e}$ is uniformly random in both the simulation and a real execution. Also, it is clear that $\psi_i^{(j)} = \mathbf{M}_\mathbf{e}\zeta_i^{(j)} + \chi_i^{(j)}$ for all $i \in [\tau]$ and $j \in T$. Thus, we need only show that $\mathbf{T}$ is statistically close to $\mathbf{M}_\mathbf{e}\mathbf{Z} + \mathbf{X}$. In the simulation, each entry of $\mathbf{T}$ is a random number bounded by $2^\tau mB$. In a real execution, each entry of $\mathbf{M}_\mathbf{e}\mathbf{Z}$ is bounded by mB. Furthermore, each entry of $\mathbf{X}$ is a random number bounded by $2^\tau mB$, which is a super-polynomially larger interval since $\tau = \omega(\log \lambda)$. Thus, the distributions of $\mathbf{T}$ and $\mathbf{M}_\mathbf{e}\mathbf{Z} + \mathbf{X}$ are statistically close.

t-Robustness: Perfect completeness follows from the fact that if each entry in $\mathbf{Z}$ is bounded by B, then each entry in $\mathbf{M}_\mathbf{e}\mathbf{Z}$ is bounded by mB and since each entry in $\mathbf{X}$ is bounded by $2^\tau mB$, then each entry in $\mathbf{M}_\mathbf{e}\mathbf{Z} + \mathbf{X}$ is bounded by $(2^\tau + 1)mB$. Now suppose an honest player outputs 1. t-robustness of π_{Check} guarantees that all shares are d-consistent. At reconstruction, players also implicitly verify that all shares in the reconstruction of $\mathbf{M}_\mathbf{e}\mathbf{Z} + \mathbf{X}$ are d-consistent. Let $\mathbf{T}$ be the reconstructed value. Recall that we chose $N - t > d$ so that the shares of the honest players uniquely determine a degree-d polynomial, and thus, uniquely determine the shares of the corrupt players. These must be the ones obtained by running the protocol honestly. Thus, there exists $\mathbf{Z} \in \mathbb{Z}_q^{m\times k}$ such that $\mathbf{T} = \mathbf{M}_\mathbf{e}\mathbf{Z} + \mathbf{X}$, namely, those corresponding to the shares held by the players.

Now, suppose an honest player accepted for two different challenges, $\mathbf{e} \neq \mathbf{e}'$, and let $\mathbf{T}, \mathbf{T}'$ be the values reconstructed at the end for each of the challenges. Then $\mathbf{T} - \mathbf{T}' = (\mathbf{M}_\mathbf{e} - \mathbf{M}_{\mathbf{e}'})\mathbf{Z}$. Let h be the largest index such that $e_h \neq e'_h$, and look at the square $\tau \times \tau$ matrix that results from only taking rows $h - \tau + 1$ to τ (included) of $(\mathbf{M}_\mathbf{e} - \mathbf{M}_{\mathbf{e}'})$. This is an upper triangular matrix with non-zero diagonal entries and we can therefore solve for $\mathbf{Z}$ from the bottom up. At each level ℓ of the recursion however, the value in question is a sum (or difference) of ℓ values, so knowing that each value is bounded by $(2^\tau + 1)mB$ means that values at the last level of the recursion, the top row of $\mathbf{Z}$, can be as large as $2^\tau \cdot (2^\tau + 1)mB < 2^{2\tau+1}mB$. Thus, this is the bound we can guarantee.

□

Claim A.2. *The subroutine π_{VerPos} described in Figure 3 is statistically t-private in the presence of a semi-honest adversary and t-robust, for $t < N/2$.*

Proof. We show statistically t-privacy in the presence of a semi-honest adversary and t-robustness, for $t < N/2$.

t-Privacy: Let $T \subset [N]$. We describe the simulator $\texttt{Sim}(T, x, R_T(x, w))$. Because we assume a semi-honest adversary, we assume $R(x, w) = 1$, and simply write $\texttt{Sim}(T, x)$. We describe the simulator in Figure 10.

Simulator $\texttt{Sim}(T, x)$ for $\pi_{\textbf{VerPos}}$

1. Let $\zeta_1^{(j)}, \ldots, \zeta_m^{(j)}$ be the input to player $\mathcal{P}_j$.
2. For $i \in [m]$ chooses random shares $\upsilon_i^{(j)}, \varphi_i^{(j)}, \omega_i^{(j)}, \psi_i^{(j)} \leftarrow \mathbb{Z}_q$ for all $j \in T$ as the shares for the corrupted parties of $\widetilde{\mathbf{U}}_i, \widetilde{\mathbf{V}}_i, \widetilde{\mathbf{W}}_i, \widetilde{\mathbf{Y}}_i$, respectively.
3. Simulates the protocol on these shares to obtain "shares" $\alpha_i^{(j)} \in \mathbb{Z}_q$ of $\mathbf{0} \in \mathbb{Z}_q^k$.
$$\alpha_i^{(j)} = \zeta_i^{(j)} - \left(\upsilon_i^{(j)}\right)^2 - \left(\varphi_i^{(j)}\right)^2 - \left(\omega_i^{(j)}\right)^2 - \left(\psi_i^{(j)}\right)^2$$
4. For $i \in [m]$, let P_i be a degree $d \geq t + k - 1$ polynomial that is consistent with secrets $\mathbf{0} \in \mathbb{Z}_q^k$ and share $\alpha_i^{(j)}$ corresponding to player $\mathcal{P}_j$ for all $j \in T$.
5. For $j \notin T$, let $\alpha_i^{(j)} = P_i(j)$.
6. Runs the simulators for π_{Check} and π_{VerBnd}.
7. Outputs the output of the simulators for π_{Check} and π_{VerBnd}, as well as all the shares $\alpha_i^{(j)}$ for $i \in [m], \jmath \in [N]$.

Figure 10: Simulator $\texttt{Sim}(T, x)$ for π_{VerPos}

Lemma 2.5 and Claim A.1 guarantee that the outputs of the simulators of π_{Check} and π_{VerBnd} are statistically close to the view of the cheating parties in a real execution of the protocol. Thus, we need only worry about the shares output by the simulator. But by properties of pack secret-sharing, these clearly have the same distribution as those output in a real execution (random subject to lying on a d-degree polynomial corresponding to secret $\mathbf{0}$).

t-Robustness: Perfect completeness of π_{VerBnd}, as well as Lagrange's theorem (see e.g. [FR06]) and the properties of packed secret-sharing described in Section 2.3 guarantee perfect completeness of π_{VerPos}. Now suppose an honest player outputs 1. t-robustness of π_{Check} guarantees that all shares distributed by $\mathcal{I}$ in Step 2 are $d/2$-consistent, and t-robustness of π_{VerBnd} guarantess that each of the entries in $\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}, \widetilde{\mathbf{W}}, \widetilde{\mathbf{Y}}$ is bounded by $B' < \sqrt{q/8}$ so we are guaranteed that each entry in $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{Y}$ is bounded by $q/8$, and therefore adding the 4 squares does not result in overflow modulo q. At reconstruction, players also implicitly verify that all shares in the construction of $\mathbf{0}$ are d-consistent. Furthermore, recall that we chose $N - t > d$ so that the shares of the honest players uniquely determine a degree-d polynomial, and thus, uniquely determine the shares of the corrupt players. These must be the ones obtained by running the protocol honestly. Thus, for all $i \in [m], j \in [k]$, there exist $u_{ij}, v_{ij}, w_{ij}, y_{ij}$, namely, those corresponding to the shares received by the players in Step 2, such that $z_{ij} = u_{ij}^2 + v_{ij}^2 + w_{ij}^2 + y_{ij}^2 < q/2$ and therefore $z_{ij} > 0$.

$\square$

$\square$