Unbounded Multi-Party Computation from Learning with Errors

Prabhanjan Ananth[1], Abhishek Jain[2], Zhengzhong Jin[2], and Giulio Malavolta[3]

[1]University of California, Santa Barbara
[2]Johns Hopkins University
[3]Max Planck Institute for Security and Privacy

Abstract

We consider the problem of round-optimal *unbounded MPC*: in the first round, parties publish a message that depends only on their input. In the second round, any subset of parties can jointly and securely compute any function f over their inputs in a single round of broadcast. We do not impose any a-priori bound on the number of parties nor on the size of the functions that can be computed.

Our main result is a semi-malicious two-round protocol for unbounded MPC in the plain model from the hardness of the standard learning with errors (LWE) problem. Prior work in the same setting assumes the hardness of problems over bilinear maps. Thus, our protocol is the first example of unbounded MPC that is post-quantum secure.

The central ingredient of our protocol is a new scheme of attribute-based secure function evaluation (AB-SFE) with *public decryption*. Our construction combines techniques from the realm of homomorphic commitments with delegation of lattice basis. We believe that such a scheme may find further applications in the future.

1 Introduction

A multi-party computation (MPC) protocol [20] allows a set of n mutually distrustful parties to evaluate any circuit C over their inputs $(x_1, \dots, x_n)$, while leaking nothing beyond the circuit output $C(x_1, \dots, x_n)$. MPC is one of the pillars of modern cryptography and the study of its round complexity (and the necessary assumptions) has motivated a large body of research. A series of recent works has established that two rounds are necessary and sufficient to securely compute any function, under a variety of cryptographic assumptions [16, 24, 30, 18, 9, 19].

A recent line of work [3, 8, 10] focuses on constructing round-optimal MPC with *reusable* first message, i.e. where the first message of the MPC can be reused an unbounded number of times for computing different functions over the committed inputs. However, out of these works only [10] achieves the "dream version" of two round MPC, i.e. an MPC that simultaneously satisfies *all* of the following properties:

- No trusted setup is required.
- In the first round, each party publishes a first message that depends only on their input and does *not* depend on the number of parties nor on the size of the circuit being evaluated.
- In the second round, any subset of parties can evaluate a circuit C over their first messages. The output can be publicly reconstructed given all the second messages.

- The second round can be repeated arbitrarily many times (with different circuits and different sets of parties), without the need to recompute a first message. Parties can join the system at any time by posting a first message.

Throughout this work, we refer to such an MPC protocol as *unbounded MPC*.

Among all works on round-optimal protocols, only [10] achieves the notion of truly unbounded MPC without the need for a trusted setup. In particular, the works of [3, 8] fall short in satisfying this notion because they impose a bound on the number of participants that needs to be fixed once and for all in the first round and needs to be shared across all parties. The earlier work of [30] does not suffer from this limitation, but requires a trusted setup.

The work of [10] assumes the hardness of standard problems over bilinear maps. While the veracity of such assumptions is well-established in the classical settings, the lurking threat of quantum computing renders such a solution immediately insecure in the presence of a scalable quantum machine. This motivates us to ask the following question:

Can we construct unbounded MPC from Learning with Errors (LWE)?

1.1 Our Results

We consider the problem of unbounded MPC with security against semi-malicious adversaries in the dishonest majority setting. In our communication model, parties publish their first message through a broadcast channel which is immediately delivered to all participants. At any point in time, any subset S of participants (with a dishonest majority) can gather together and evaluate a circuit C over their inputs $(x_1, \dots, x_{|S|})$ in a *single round* of broadcast. The output $C(x_1, \dots, x_{|S|})$ can then be publicly reconstructed from the messages of all parties. This phase can be repeated arbitrarily many times without having to re-initialize the first message (i.e. the first message is reusable). We do not impose any a-priori bound on the number of participants nor on the size of the circuits. We prove the following theorem:

Theorem 1.1 (Informal)**.** *If the learning with errors (LWE) problem is hard, then there exists a two-round unbounded MPC in the plain model.*

By additionally assuming the quantum hardness of LWE, we obtain the first post-quantum secure protocol for (semi-malicious) unbounded MPC in two rounds. Our main technical ingredient is a new construction of attribute-based secure function evaluation (AB-SFE) [28] where the output can be *publicly reconstructed* at the end of the second round. On a technical level, our scheme combines the homomorphic commitment scheme from [23] with techniques to delegate a lattice basis. We believe that such a scheme may find further applications in the future.

2 Technical Overview

In the following, we summarize the main technical innovations of our work. This outline can be roughly split in three components: First we introduce the notion of AB-SFE [28] with public decryption and we recall the security properties that we want to guarantee. Then we show an instantiation of AB-SFE with public decryption from LWE, building on the construction of homomorphic commitments from [23]. Finally, we show how AB-SFE functions as the main ingredient (alongside garbled circuits) for constructing unbounded MPC.

2.1 AB-SFE with Public Decryption

We begin by recalling the notion of AB-SFE [28]. AB-SFE was introduced in the context of designated-verifier non-interactive zero-knowledge proof to obtain constructions from new assumptions. However the work of [28] focused on the notion where decrypting a message requires a *secret state* (that might leak some information about the attribute). Here we augment the syntax of AB-SFE with a *public decryption* procedure. For the purpose of our work, it is going to be useful to cast this primitive as a two-party protocol between an "authority" and a "sender." The interaction proceeds as follows:

- **Key Generation:** On input an attribute x, the authority locally runs a setup algorithm $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and generates a secret/public key pair $(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, x)$.[1]
- **Encryption:** Given the public key pk (generated as above), a circuit C and a message μ, the sender computes a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, C, \mu)$.
- **Decryption Hint:** To enable public decryption, the authority crafts a circuit-specific decryption hint $\mathsf{sk}_C \leftarrow \mathsf{Hint}(\mathsf{msk}, C)$.
- **Public Decryption:** Anyone who possesses the ciphertext ct and the decryption hint sk_C can recover the message μ by running $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct})$. The procedure succeeds if and only if $C(x) = 1$.

One way to interpret this primitive is as a secure two-party computation protocol where the interaction consists only of two rounds and where only one party speaks in the first round. Looking ahead, this latter property is going to be crucial to achieve unbounded secure MPC, since it will allow *multiple (unbounded) parties to simultaneously play the role of the sender.*

Security of AB-SFE. As for the security of AB-SFE we define two properties: (1) We require that nothing beyond $C(x)$ is revealed about the attribute x. This requirement must hold even for polynomially many circuits $(C_1, \dots, C_q)$ and in the presence of the corresponding decryption hints $(\mathsf{sk}_{C_1}, \dots, \mathsf{sk}_{C_q})$, for any polynomial q. (2) We require that for all circuits C such that $C(x) = 0$ it holds that

$$\mathsf{Enc}(\mathsf{pk}, C, \mu_0) \approx \mathsf{Enc}(\mathsf{pk}, C, \mu_1)$$

are computationally indistinguishable. This is required to hold *even if the distinguisher is given the random coins used in the key generation procedure.* In other words, if the circuit outputs 0, even the key authority should not be able to learn the message of the sender. This is in stark contrast with the standard attribute-based encryption settings [33, 25] where typically semantic security does *not* hold against a corrupted authority.

2.2 AB-SFE from Learning with Errors

The problem of constructing AB-SFE was considered in [28] where they obtained schemes from a variety of assumptions in the private decryption settings, based on 2-round oblivious transfer. However, none of their schemes support *public decryption* (without adding an extra round of interaction).

In this work, we take a different route. Our starting point is the fully homomorphic commitment scheme from [23], which we briefly recall in the following.

[1]Note that we could have merged the Setup and the KeyGen algorithms in a single subroutine, however we refrained to do so in order to match the original syntax from [28].

Homomorphic Commitments. The commitment key is a uniform matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and committing to a multi-bit string $(x_1, \ldots, x_u)$ corresponds to the computation of a set of

$$\mathbf{C}_i = \mathsf{Com}(\mathbf{A}, x_i; \mathbf{R}_i) = \mathbf{A} \cdot \mathbf{R}_i + x_i\mathbf{G}$$

where $\mathbf{R}_i \leftarrow \{0,1\}^{m\times m}$ and $\mathbf{G}$ is the gadget matrix from [29]. Here $\mathbf{R}_i$ is a low-norm vector and plays the role of the decommitment. In [23] it is shown that one can homomorphically evaluate any (depth-bounded) circuit C over committed value and still obtain a well-formed commitment $\mathbf{C}_C$. The exact details of the algorithm are irrelevant for the purpose of this overview, except for the fact that one can define a (deterministic) homomorphic computation over the decommitments and obtain a low-norm vector $\mathbf{R}_{C,x}$, which is a valid decommiment for $\mathbf{C}_C$.

At this point it is instructive to take a step back and think how we could implement AB-SFE if we had a general-purpose witness encryption [17] scheme. A witness encryption scheme, associated with a NP language, consists of an encryption and a decryption algorithm: Anyone can encrypt their message μ under an NP instance and the decryption algorithm can obtain μ using the witness to this instance. We use witness encryption as follows: The sender encrypts μ under the instance $\mathbf{A} \cdot \mathbf{R}_{C,x} + C(x)\mathbf{G}$ which is obtained by homomorphically evaluating upon the commitments using the circuit C. The authority releases the decomitment $\mathbf{R}_{C,x}$ as witness which would then allow anyone to recover μ if and only if $C(x) = 1$. Temporarily glossing over the fact that $\mathbf{R}_{C,x}$ might leak some information about x, we are going to show how to implement this idea without resorting to the power of general-purpose witness encryption.

Computing Hints via Basis Delegation. Our first observation is that, when $C(x) = 1$, the matrix $\begin{bmatrix}\mathbf{A} & \mathbf{C}_C\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{A}\mathbf{R}_{C,x} + \mathbf{G}\end{bmatrix}$ matches the construction of lattice trapdoor in [29]. Hence, $\mathbf{R}_{C,x}$ allows us to compute a short basis (a trapdoor) for the dual lattice spanned by $\begin{bmatrix}\mathbf{A} & \mathbf{C}_C\end{bmatrix}$. Following [29], such a trapdoor $\mathbf{T}$ can be efficiently computed in the following way

$$\mathbf{T} = \begin{bmatrix}\mathbf{I} & -\mathbf{R}_{C,x} \\ \mathbf{0} & \mathbf{I}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{I} & \mathbf{0} \\ -\mathbf{G}^{-1}[\mathbf{A}] & \mathbf{T}_{\mathbf{G}}\end{bmatrix}$$

where $\mathbf{T}_{\mathbf{G}}$ is a short basis for the lattice $\Lambda_q^{\perp}(\mathbf{G})$, which is publicly computable. At this point it is tempting to view $\begin{bmatrix}\mathbf{A} & \mathbf{C}_C\end{bmatrix}$ as the public-key of the witness encryption and $\mathbf{T}$ as the witness. After all, $\mathbf{T}$ has low norm if and only if $\mathbf{R}_{C,x}$ does, which implies that $\mathbf{R}_{C,x}$ is a valid decommitment for $\mathbf{C}_C$.

However we are not yet done. The adversary receives $\mathbf{R}_{C,x}$, for multiple circuits, where each decommitment is a deterministic function of the decomitments $(\mathbf{R}_1, \ldots, \mathbf{R}_u)$ and enough number of such decommitments will leak some information about x. Recall that we are interested in the public decryption setting, which would require us to publicly release $\mathbf{T}$, which is again a deterministic function of $\mathbf{R}_{C,x}$.

Our next idea is to *randomize* the trapdoor $\mathbf{T}$ using the basis delegation procedure of [13]. In the literature, this process is also referred to as *SampleRight*. First we add a uniformly sampled matrix $\widehat{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n\times 2m}$ to the public paramenters. Given the trapdoor $\mathbf{T}$ for $\begin{bmatrix}\mathbf{A} & \mathbf{C}_C\end{bmatrix}$, the inverse sampling algorithm allows us to probabilistically sample a short basis $\mathbf{H}$ for the lattice

$$\Lambda_q^{\perp}\left(\begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A} & \mathbf{C}_C\end{bmatrix}\right)$$

and $\mathbf{H}$ carries no information about $\mathbf{T}$. At this point we have all ingredients to instantiate our witness encryption: After recomputing $\mathbf{C}_C$ homomorphically, the encryptor parses

$$\mathbf{A}'' = \begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A} & \mathbf{C}_C\end{bmatrix}$$

as a public matrix for LWE and encodes μ as $(\mathbf{s} \cdot \mathbf{A}'' + \mathsf{noise}, \mathsf{Ext}(\mathbf{s}, \mathbf{r}) \oplus \mu, \mathbf{r})$, where $\mathbf{s}$ is a secret vector sampled from discrete Gaussian and Ext is an extractor. The decryption hint $\mathbf{H}$ can be computed from $\mathbf{R}_{C,x}$ as described above and allows anyone to recover $\mathbf{s}$ and hence decrypts μ, since $\mathbf{H}$ is a short basis for $\Lambda_q^{\perp}(\mathbf{A}'')$. To prove the sementic security, we observe that $\mathbf{s}\mathbf{A}'' + \mathsf{noise} = \begin{bmatrix}\mathbf{s} \cdot \begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}\end{bmatrix} & \mathbf{s} \cdot \mathbf{A}\mathbf{R}_{C,x}\end{bmatrix} + \mathsf{noise}$. Then, by noise flooding, the term $\mathbf{s} \cdot \mathbf{A}\mathbf{R}_{C,x} + \mathsf{noise}$ can be simulated by only using $\mathbf{s} \cdot \begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}\end{bmatrix} + \mathsf{noise}$. While the later computationally hides $\mathbf{s}$ by LWE assumption. Hence, the sender's security follows from the randomness extraction of Ext.

To see why we achieve security against a corrupted sender, we first switch from using a trapdoor for $\begin{bmatrix}\mathbf{A} & \mathbf{C}_C\end{bmatrix}$ to generate the matrix $\mathbf{R}_{C,x}$ to instead use a trapdoor for $\widehat{\mathbf{A}}$ (using a process referred to as *SampleLeft*)[2]; this switch is statistically indistinguishable and follows from the standard lattice trapdoor lemmas. We do this switch for every circuit. Once we do this, we then invoke leftover hash lemma to instead generate the commitment as $\mathbf{U}_\mathbf{i} + x_i\mathbf{G}$, where $\mathbf{U}_\mathbf{i}$ is generated uniformly at random. At this point, the input of the receiver is information-theoretically hidden from the sender.

Achieving Security Against Semi-Malicious Receivers. One downside of the above construction is that it ony achieves semi-honst security and in particular is not semi-malicious secure. For example, a semi-malicious receiver can generate the matrix $\mathbf{A}$ with a trapdoor, and then learn the message μ regardless of whether $C(x) = 1$ or not. To further achieve the semi-malcious security, our idea is to leverage the following observation in [11]: if the lattice $\Lambda_q\left(\begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}\end{bmatrix}\right)$ has a basis with $n/2$ short vectors, then the encoding $\mathbf{s} \mapsto \mathbf{s} \cdot \begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}\end{bmatrix} + \mathsf{noise}$ is lossy for $\mathbf{s}$. Based on this observation, we sample the matrices $\widehat{\mathbf{A}}$ and $\mathbf{A}$ in a structured way such that they always have a basis with $n/2$ short vectors. Now, when $C(x) = 0$, the public matrix $\mathbf{A}''$ becomes

$$\mathbf{A}'' = \begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A} & \mathbf{A} \cdot \mathbf{R}_{C,x}\end{bmatrix}.$$

Then the lattice $\Lambda_q^{\perp}(\mathbf{A}'')$ also has a basis with $n/2$ short vectors. Then we can use the aforementioned observation in [11] to prove (statistical) semantic security.

We sample $\begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}\end{bmatrix}$ with $n/2$ short basis as

$$\begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}\end{bmatrix} = \begin{bmatrix}\widehat{\mathbf{B}} & \mathbf{B} \\ \mathbf{S} \cdot \widehat{\mathbf{B}} + \widehat{\mathbf{E}} & \mathbf{S} \cdot \mathbf{B} + \mathbf{E}\end{bmatrix}$$

where $\widehat{\mathbf{A}}, \mathbf{A}$ are two matrices consisting of LWE instance, and hence are pseudorandom. On the other hand, the lattice $\Lambda_q\left(\begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}\end{bmatrix}\right)$ has a basis $\begin{bmatrix}\widehat{\mathbf{B}} & \mathbf{B} \\ \widehat{\mathbf{E}} & \mathbf{E}\end{bmatrix}$, where the last $n/2$ rows are short vectors. Then the above argument follows, and we achieve (statistical) sender's security against semi-malicious receiver.

2.3 From AB-SFE with Public Decryption to Unbounded MPC

We are now ready to show how AB-SFE with public decryption readily gives us a construction of unbounded MPC.

Building Blocks. In addition to AB-SFE with public decryption, we are going to assume the existence of any semi-malicious secure two-round MPC, denoted by mpc, such as the protocols proposed in [9, 19]. We note that we do not place any additional restrictions on mpc: For instance, it need not guarantee any reusability property and moreover, the total number of parties in the MPC protocol can be fixed before the

[2]In the technical sections, instead of using the terms SampleLeft and SampleRight, we use the algorithm GenSamplePre that captures the functionality of both these algorithms.

first round message. Furthermore we are going to make use of garbled circuits [34]. For the reader unfamiliar with the notion, a garbling scheme allows one to compute a garbled version of a circuit C together with set of label pairs $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1})$. Given an input z, its encoding consists of the labels corresponding to its bit representation $(\mathsf{lab}_{1,z_1}, \ldots, \mathsf{lab}_{|z|,z_{|z|}})$ and security requires that nothing is revealed about z, besides the output of the computation $C(z)$.

It is also going to be convenient to consider an augmented notion of AB-SFE, that we denote by 2AB-SFE, following the convention from [22]. A 2AB-SFE with public decryption is identical to AB-SFE with public decryption except that the encryption algorithm takes as input two messages (μ_0, μ_1) and the public decryption returns μ_0 if $C(x) = 0$ and μ_1 if $C(x) = 1$. Given an AB-SFE, it is easy to construct a 2AB-SFE by just encrypting μ_0 under the complement of C.

The Unbounded MPC Protocol. We provide a simplified desciption of our unbounded MPC in the following.

- **First Message:** Given an input x_i, the first message of each party simply consists of the generation of a public key pk_i for the 2AB-SFE scheme, where the attribute is set to the input x_i.
- **Second Message:** Each party P_i is given as input set of parties S and a circuit C. First, it computes a garbled version of the circuit that takes as input S (specifying the subset of parties participating in the protocol), any first round messages $(m_1, \ldots, m_{|S|})$ of mpc and computes the i^{th} party's second round messages of mpc (the input x_i is hardwired in the computation). After it computes the garbled circuit, it then takes each pair of labels $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1})$ and computes a 2AB-SFE encryption for the corresponding participant P_j under the circuit $\Gamma_{i,j}$, defined as follows.

  $$\Gamma_{i,j} : \text{Compute the } i\text{-th bit of } m_j.$$

  Finally, for all $j = 1 \ldots |S|$ compute the decryption hints for the 2AB-SFE encryption corresponding to the circuit $\Gamma_{j,i}$.
- **Reconstruction:** The public reconstruction algorithm works by using all the decryption hints to recover all the labels, which in turn are used to evaluate the garbled circuits. This results in a set of second round messages $(p_1, \ldots, p_{|S|})$ for the underlying two-round MPC. The reconstruction algorithm then returns the result of the reconstruction procedure of the one-time MPC.

Since the first message consists only of the key of the 2AB-SFE scheme, it is clear that the resulting MPC does not impose a bound on the parties. Also note that the underlying two-round secure MPC, namely mpc, is freshly re-initialized for each second message and therefore the security of the reusable protocol is not affected. One subtlety that we ignored in the above description is that the computation of the messages for the one-time MPC is randomized and we need to ensure that the same randomness is used consistently in the first and second message for each party. This can be done routinely by adding a PRF key alongside the input and drawing all necessary random coins by evaluating the PRF on some public input.

Circuit-Size-Independent Communication. As described above, our scheme does not achieve circuit-size-independent communication. However, by using a slight variation of a compiler from [32], we can apply a generic transformation to achieve this property. In [32] it was observed that any non-compact 2-round MPC can be generically transformed into a compact one via laconic function evaluation (LFE).[3] The CRS for LFE is chosen by one of the parties and sent in the first round. In our setting, this does not work

[3] A similar transformation was also used in [2] to compress the communication complexity.

as-is since the length of CRS grows linearly in the input length of the function being computed, which is unbounded. A simple solution is to have each party send a (uniformly random) CRS whose length is only proportional to their own input length. Later, when a subset of parties want to evaluate some function, they first concatenate their CRSes to obtain a long enough CRS, then the 2-round MPC protocol to compute the LFE ciphertext, as done in [32, 2] for achieving circuit-independent communication.

In summary, by using the above observation, we can achieve both unbounded-party property and circuit-independent communication (except for a factor that depends on the depth of the circuit, due to the known construction of LFE).

2.4 Related Work

Ishai et al. [27] introduced the notion of reusable non-interactive secure computation (rNISC), where a receiver can publish a reusable encoding of its input y and any sender can enable computation of $f(x, y)$ by computing a message using input x and sending it to the receiver. This notion has subsequently been studied in many follow-up works; see, e.g., [1, 12, 6, 7, 14].

The recent work of Benhamouda and Lin [10] extends this notion to the *multiparty* setting, and refers to it as multiparty reusable NISC (mrNISC). Unlike rNISC which is primarily challenging in the malicious adversary model (from the viewpoint of black-box constructions), mrNISC is non-trivial even in the semi-honest adversary model. Unbounded MPC seeks the same goals as mrNISC; we use the former terminology to emphasize the key property that the first round messages do not depend on the number of parties or the size of the circuit or the size of the subset of parties involved in the actual computation.

3 Preliminaries

3.1 Notations

For any integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We use $\mathbb{Z}$ to denote the sets of integers, and use $\mathbb{Z}_q$ to denote $\mathbb{Z}/q\mathbb{Z}$.

For any sets $S_1, S_2, \ldots, S_n$ of integers, and any tuple $(i_1^*, i_2^*, \ldots, i_n^*) \in S_1 \times S_2 \times \cdots \times S_n$, we use the notation $(i_1^*, i_2^*, \ldots, i_n^*) + 1$ (resp. $(i_1^*, i_2^*, \ldots, i_n^*) - 1$) to denote the lexicographical smallest (resp. biggest) element in $S_1 \times S_2 \times \cdots \times S_n$ that is lexicographical greater (resp. less) than $(i_1^*, i_2^*, \ldots, i_n^*)$.

Statistical Distance. For any two discrete distributions P, Q, the statistical distance between P and Q is defined as $\mathsf{SD}(P, Q) = \sum_i \left| \Pr[P = i] - \Pr[Q = i] \right| / 2$ where i takes all the values in the support of P and Q.

Matrix Norms. For any matrix $\mathbf{A}$, let $\|\mathbf{A}\|$ be the maximum ℓ_2 norm of its columns, and $\|\mathbf{A}\|_2$ be the ℓ_2 norm of $\mathbf{A}$.

Extractors. An algorithm $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^\ell$ is a seeded strong average-case (k, ϵ)-extractor, if for any random variables X over $\{0,1\}^n$ and Z with $\widetilde{\mathrm{H}}_\infty(X|Z) \geq k$, then $\mathsf{SD}((\mathsf{Ext}(X, \mathbf{r}), \mathbf{r}, Z), (\mathbf{u}, \mathbf{r}, Z)) < \epsilon$, where $\mathbf{u} \leftarrow \{0,1\}^\ell$ and $\mathbf{r} \leftarrow \{0,1\}^r$ are sampled uniformly at random.

Theorem 3.1 (Seeded Extractor [26])**.** *For every constant $\alpha > 0$, and all positive integers n, k and $\epsilon > 0$, there exists an explicit construction of a strong (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$ with $r = O(\log n + \log(1/\epsilon))$ and $m \geq (1 - \alpha)k$.*

3.2 Lattices and LWE Assumption

Let m be an integer, a lattice is a discrete additive group in $\mathbb{R}^m$. We say that a set of linear independent vectors $\mathbf{B} = \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k\}$ is a basis of a lattice Λ, if $\Lambda = \{\mathbf{Bz} \mid \mathbf{z} \in \mathbb{Z}^k\}$. Let $\widetilde{\mathbf{B}} = (\widetilde{\mathbf{b}_1}, \widetilde{\mathbf{b}_2}, \ldots, \widetilde{\mathbf{b}_k})$ be the Gram-Schmidt basis derived from $\mathbf{B}$.

For any integer $n, m, q \geq 2$ and $\mathbb{Z}_q^{n\times m}$, we define the q-ary lattice

$$\Lambda_q(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}_q^m \mid \exists \mathbf{s} \in \mathbb{Z}^n, \mathbf{z} = \mathbf{A}^T\mathbf{s} \pmod{q}\}$$
$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}_q^m \mid \mathbf{Az} = \mathbf{0} \pmod{q}\}$$

Similarly, for any $\mathbf{y} \in \mathbb{Z}_q^n$, we define the coset $\Lambda_q^{\mathbf{y}}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}_q^m \mid \mathbf{Az} = \mathbf{y} \pmod{q}\}$.

Discrete Gaussian. For any integer n and real $s > 0$, define the Gaussian function $\rho_s : \mathbb{R} \longrightarrow \mathbb{R}^+$ of parameter s as $\rho_s(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/s^2)$. For any lattice Λ, any vector $\mathbf{c} \in \mathbb{R}^m$, and real $s > 0$, we denote $\rho_s(\Lambda + \mathbf{c}) = \sum_{\mathbf{x}\in\Lambda} \rho_s(\mathbf{x} + \mathbf{c})$. The discrete Gaussian probabilistic distribution $D_{\Lambda+\mathbf{c},s}$ is a distribution over Λ with density function $\rho_s(\mathbf{x})/\rho_s(\Lambda + \mathbf{c})$, for any $\mathbf{x} \in \Lambda$.

Theorem 3.2 (Noise Flooding [5, 21, 15, 31]). *For any $c \in \mathbb{Z}$, and real $s > 0$,* $\mathsf{SD}(D_{\mathbb{Z},s}, D_{c+\mathbb{Z},s}) < O(c/s)$.

Definition 3.3 (LWE Assumption). *Let $n = n(\lambda), m = m(\lambda), \ell = \ell(\lambda)$ be polynomials in λ, and let the modulus $q = 2^{\lambda^{O(1)}}$ be a function of λ, and $\chi = \chi(\lambda)$ be a noise distribution. The Learning with Error (LWE) assumption states that for any PPT distinguisher $\mathcal{D}$, there exists a negligible function $\nu(\lambda)$ such that*

$$\left|\Pr\left[\mathcal{D}(1^\lambda, (\mathbf{A}, \mathbf{S}\cdot\mathbf{A} + \mathbf{E})) = 1\right] - \Pr\left[\mathcal{D}(1^\lambda, (\mathbf{A}, \mathbf{U})) = 1\right]\right| \leq \nu(\lambda),$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{S} \leftarrow \mathbb{Z}_q^{\ell\times n}, \mathbf{U} \leftarrow \mathbb{Z}_q^{\ell\times m}, \mathbf{E} \leftarrow \chi^{\ell\times m}$.

Lattice Trapdoor and Preimage Sampling. For any integer $n, q, m = n\log q$, let $\mathbf{G} \in \mathbb{Z}^{n\times m}$ be the gadget matrix in [29], and let $\mathbf{G}^{-1}[\cdot] : \mathbb{Z}_q^{n\times m} \longrightarrow \{0,1\}^{m\times m}$ be the bit-decomposition function. Let $\mathbf{T}_{\mathbf{G}} \in \mathbb{Z}^{m\times m}$ be the small basis of $\Lambda_q^{\perp}(\mathbf{G})$.

Theorem 3.4 ([29], Theorem 5.1). *There is an efficient randomized algorithm* $\mathsf{TrapGen}(1^n, 1^m, q)$, *that given any integer $n \geq 1, q \geq 2$, and sufficiently large $m = O(n\log q)$, outputs a (partity-check) matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and a short basis $\mathbf{T}$ for $\Lambda_q^{\perp}(\mathbf{A})$, such that $\mathbf{A}$ is statistically close to uniform.*

Theorem 3.5 ([13], Theorem 3.3). *Let n, m, q, k be positive integers such that $q \geq 2, m \geq 2n\log_2 q$. There exists a PPT algorithm* $\mathsf{SampleBasis}$, *that on input of $\mathbf{A} \in \mathbb{Z}_q^{n\times km}$, a set $S \subseteq [n]$ and $\mathbf{B}_S$ is a basis for $\Lambda_q^{\perp}(\mathbf{A}_S)$, and an integer $L \geq \|\widetilde{\mathbf{B}}_S\| \cdot \sqrt{km} \cdot \omega(\sqrt{\log km})$, outputs $\mathbf{B} \leftarrow \mathsf{SampleBasis}(\mathbf{A}, \mathbf{B}_S, S, L)$ such that for an overwhleming fraction of $\mathbf{A}$, $\mathbf{B}$ is a basis for $\Lambda_q^{\perp}(\mathbf{A})$ with $\|\widetilde{\mathbf{B}}\| \leq L$ (with overhelming probability). Furthermore, the distribution of $\mathbf{B}$ only depends on $\mathbf{A}$ and L.*

3.3 Garbling Scheme

A garbling scheme is a pair of algorithms $(\mathsf{Garble}, \mathsf{Eval})$, which works as follows.

- $\mathsf{Garble}(1^\lambda, C)$: The garbling algorithm takes as input a security parametere λ, and a circuit C with input length ℓ_{in} and output length ℓ_{out}. Then it outputs a garbled circuit $\widetilde{C}$ and some labels $\mathsf{lab} = \{\mathsf{lab}_{b,i}\}_{b\in\{0,1\}, i\in[\ell_{\mathsf{in}}]}$.

- $\mathsf{Eval}(\widetilde{C}, \mathsf{lab}_x)$: For any $x \in \ell_{\mathsf{in}}$, let lab_x denote $\{\mathsf{lab}_{x_i,i}\}_{i\in[\ell_{\mathsf{in}}]}$. On input $\widetilde{C}$ and lab_x, it outputs a y.

We require a garbling scheme to satisfy the following properties.

- **Correctness:** For any circuit $C : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$, and any input $x \in \{0,1\}^{\ell_{\mathsf{in}}}$, we have

$$\Pr\left[(\widetilde{C}, \mathsf{lab}) \leftarrow \mathsf{Garble}(1^\lambda, C), y \leftarrow \mathsf{Eval}(\widetilde{C}, \mathsf{lab}_x) : y = C(x)\right] = 1$$

- **Simulation Security:** There exists a simulator Sim such that for any n.u. PPT distinguisher $\mathcal{D}$, there exists a negligible function $\nu(\lambda)$ such that

$$\Bigg| \Pr\left[(\widetilde{C}, \mathsf{lab}) \leftarrow \mathsf{Garble}(1^\lambda, C) : \mathcal{D}(1^\lambda, \widetilde{C}, \mathsf{lab}_x) = 1\right] - \Pr\left[(\overline{C}, \overline{\mathsf{lab}}) \leftarrow \mathsf{Sim}(1^\lambda, C(x)) : \mathcal{D}(1^\lambda, \overline{C}, \overline{\mathsf{lab}}) = 1\right] \Bigg| < \nu(\lambda)$$

3.4 Semi-Malicious 2-round MPC in Plain Model

A (one-time useable, selective secure) semi-malicious 2-round MPC in the plain model is a tuple of algorithms $(\mathsf{Round}_1, \mathsf{Round}_2, \mathsf{Rec})$, which work as follows.

There are N parties who want to jointly compute $f(x_1, x_2, \ldots, x_N)$, where x_i is the input of i-th party.

- **Round 1:** For each $i \in [N]$, the i-th party sets fresh random coins r_i, and executes $\mathsf{msg}_i \leftarrow \mathsf{Round}_1(1^\lambda, x_i, f; r_i)$.
- **Round 2:** For each $i \in [N]$, the i-th party executes $p_i \leftarrow \mathsf{Round}_2(x_i, r_i, \{\mathsf{msg}_j\}_{j\in[N]})$.
- **Output Recovery:** Any one with $\{p_i\}_{i\in[N]}$ executes $y \leftarrow \mathsf{Rec}(\{p_i\}_{i\in[N]})$.

We require the protocol to satisfy the following property.

- **Semi-Malicious Simulation Security:** There exists a simulator Sim such that, for any input $\{x_i\}_{i\in[N]}$, for any subset of honest parties $H \subseteq [N]$, and any dishonest parties' random coins $\{r_i\}_{i\in[N]\setminus H}$, any PPT distinguisher $\mathcal{D}$, there exists a negligible function $\nu(\lambda)$ such that for any sufficiently large λ,

$$\Bigg| \Pr\left[\substack{\forall i\in H, r_i \leftarrow \{0,1\}^*, \forall i\in[N], \mathsf{msg}_i = \mathsf{Round}_1(1^\lambda, x_i; r_i), \\ p_i = \mathsf{Round}_2(x_i, r_i, \{\mathsf{msg}_j\}_{j\in[N]})} : \mathcal{D}(1^\lambda, \{\mathsf{msg}_i, p_i\}_{i\in[N]}) = 1\right] - \Pr\left[\mathcal{D}(1^\lambda, \mathsf{Sim}(1^\lambda, H, \{x_i, r_i\}_{i\notin H}, f, f(\{x_i\}_{i\in[N]}))) = 1\right] \Bigg| \le \nu(\lambda)$$

Here, without loss of generality, we assume the Round_1 and Round_2 use the same random coins.

3.5 Homomorphic Commitment

A homomorphic commitment scheme is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Eval})$, with the following syntax.

- $\mathsf{Gen}(1^\lambda)$: A CRS generation algorithm that takes as input a security parameter λ, and it outputs a common random string crs.

- $\mathsf{Com}(\mathrm{crs}, \mu; r)$: A commitment algorithm that takes as input the CRS crs, a message $\mu \in \{0,1\}$, and randomness r, it outputs a commmiment c.
- $\mathsf{Eval}(C, (c_1, c_2, \ldots, c_u))$: The (fully) homomorphic evaluation algorithm Eval takes as input a circuit C, and some commitments $c_1, c_2, \ldots, c_u$, and it outputs an evaluated commitment $\mathsf{Com}(C(x); r_{C,x})$, where $x = (x_1, x_2, \ldots, x_u)$ is the message that $c_1, c_2, \ldots, c_u$ committed. Furthermore, the randomness $r_{C,x}$ can be efficiently computed from the randomenss used to compute $c_1, c_2, \ldots c_u$ and x.

We require it to satisfy the following properties.

Statistical Hiding. There exists a negligible function $\nu(\lambda)$ such that,

$$\mathsf{SD}((\mathrm{crs}, \mathsf{Com}(\mathrm{crs}, 0)), (\mathrm{crs}, \mathsf{Com}(\mathrm{crs}, 1))) \leq \nu(\lambda),$$

where the randomness is over the CRS crs $\leftarrow \mathsf{Gen}(1^\lambda)$ and the randomness used to compute the commitment.

Construction. Let $n = n(\lambda)$ be a polynomial in λ, $q = q(\lambda)$ be a function of λ, and $m = n \log q$.

- $\mathsf{Gen}(1^\lambda)$: It samples $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ uniformly at random, and output crs $= \mathbf{A}$.
- $\mathsf{Com}(\mathrm{crs} = \mathbf{A}, \mu \in \{0,1\}; \mathbf{R})$: It outputs a commitment $\mathbf{C} = \mathbf{A}\mathbf{R} + \mu\mathbf{G}$.
- $\mathsf{Eval}(C, (\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_u))$: For each gate in the circuit C, the homomorphic evaluation algorithm performs the following:
  - For each addition gate, let the commitment of the input wires to be $\mathbf{C}_l, \mathbf{C}_r$, it computes the commitment for the output wire as follows.
$$\mathbf{C}_o = \mathbf{C}_l + \mathbf{C}_r$$
  - For each multiplication gate, let the commitment of the input wires to be $\mathbf{C}_l, \mathbf{C}_r$, it computes the commitment for the output wire as follows.
$$\mathbf{C}_o = \mathbf{C}_l \mathbf{G}^{-1}[\mathbf{C}_r]$$

Lemma 3.6 (Bound on Homomorphic Evaluation). *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix, $x = (x_1, x_2, \ldots, x_u)$ be a binary string, and C be a boolean circuit of depth d. Let*

$$\mathbf{C} = \mathsf{Eval}(C, (\mathsf{Com}(\mathbf{A}, x_1; \mathbf{R}_1), \mathsf{Com}(\mathbf{A}, x_2; \mathbf{R}_1), \ldots, \mathsf{Com}(\mathbf{A}, x_u; \mathbf{R}_u))),$$

where $x_1, x_2, \ldots, x_u \in \{0,1\}$, and $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_u \in \{0,1\}^{m \times m}$. Then there exists a $\mathbf{R}_{C,x}$ that can be efficiently computed from $x_1, x_2, \ldots, x_u$ and $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_u$ such that $\mathbf{C} = \mathsf{Com}(\mathbf{A}, C(x_1, x_2, \ldots, x_u); \mathbf{R}_{C,x})$ and $\|\mathbf{R}_{C,x}\|_2 \leq 2^{O(d \log m)}$.

Proof. We analysis for each gate. For each addition gate, if $\mathbf{C}_l = \mathbf{A}\mathbf{R}_l + \mu_l\mathbf{G}$, and $\mathbf{C}_r = \mathbf{A}\mathbf{R}_r + \mu_r\mathbf{G}$, then $\mathbf{C}_o = \mathbf{A}(\mathbf{R}_l + \mathbf{R}_r) + (\mu_r + \mu_l)\mathbf{G}$. Hence, if we let $\mathbf{R}_o = \mathbf{R}_l + \mathbf{R}_r$, then $\|\mathbf{R}_o\|_2 \leq \|\mathbf{R}_l\|_2 + \|\mathbf{R}_r\|_2$.

For each multiplication gate, $\mathbf{C}_o = \mathbf{C}_l\mathbf{G}^{-1}[\mathbf{C}_r] = \mathbf{A}(\mathbf{R}_l\mathbf{G}^{-1}[\mathbf{C}_r] + \mu_r\mathbf{R}_r) + \mu_l\mu_r\mathbf{G}$. Let $\mathbf{R}_o = \mathbf{R}_l\mathbf{G}^{-1}[\mathbf{C}_r] + \mu_r\mathbf{G}_r$. Hence, $\|\mathbf{R}_o\|_2 \leq m\|\mathbf{R}_l\|_2 + \|\mathbf{R}_r\|_2$.

Hence, by induction on the depth of the circuit, we have that $\|\mathbf{R}_{C,x}\|_\infty \leq m^{O(d)}$. □

4 Secure Function Evaluation with Public Decryption

4.1 Definition

An AB-SFE with public decryption is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Hint}, \mathsf{Dec})$, with the following syntax.

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, it outputs a common random string crs.
- $\mathsf{KeyGen}(\mathsf{crs}, x)$: On input the crs, and a binary string x, it outputs a public key pk and a master secret key msk.
- $\mathsf{Enc}(\mathsf{pk}, C, \mu)$: On input the public key pk, a boolean circuit $C : \{0,1\}^{|x|} \to \{0,1\}$, and a message $\mu \in \{0,1\}$, it outputs a ciphertext ct.
- $\mathsf{Hint}(\mathsf{msk}, C)$: On input the master secret key, and the circuit C, output a hint sk_C.
- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct})$: On input a hint sk_C, and a ciphertext ct, it outputs a message μ'.

We require the AB-SFE to satisfy the following properties.

- **Correctness.** For any binary string x, any circuit $C : \{0,1\}^{|x|} \to \{0,1\}$ with $C(x) = 1$, and any message $\mu \in \{0,1\}$, there exists a negligible function $\nu(\lambda)$ such that for any sufficiently large λ,

$$\Pr\left[\substack{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk},\mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs},x), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk},C,\mu) \\ \mathsf{sk}_C \leftarrow \mathsf{Hint}(\mathsf{msk},C), \mu' \leftarrow \mathsf{Dec}(\mathsf{sk}_C,\mathsf{ct})} : \mu = \mu'\right] \geq 1 - \nu(\lambda)$$

- **Statistical Indistinguishability of Public Keys.** There exists a negligible function $\nu(\lambda)$ such that, with $1 - \mathsf{negl}(\lambda)$ probability over the randomness of $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, for any x_0, x_1 with $|x_0| = |x_1|$, and

$$\mathsf{SD}\left(\mathsf{pk}_0, \mathsf{pk}_1\right) \leq \nu(\lambda)$$

  where pk_b is generated by $\mathsf{KeyGen}(\mathsf{crs}, x_b)$ for $b \in \{0,1\}$.

- **Simulation of Hints.** There exists a negligible function $\nu(\lambda)$, a PPT crs generating function $\overline{\mathsf{Setup}}(1^\lambda)$ such that, for any n.u. PPT adversary $\mathcal{A}$,

$$\left|\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}(1^\lambda, \mathsf{crs}) = 1\right] - \Pr\left[(\overline{\mathsf{crs}}, \mathsf{tr}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda) : \mathcal{A}(1^\lambda, \overline{\mathsf{crs}}) = 1\right]\right| \leq \nu(\lambda).$$

  Furthermore, there exists a PPT simulator Sim such that, for any input string x, any circuit C, let $(\overline{\mathsf{crs}}, \mathsf{tr}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$, $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\overline{\mathsf{crs}}, x)$,then

$$\mathsf{SD}\left(\mathsf{Hint}(\mathsf{msk}, C), \mathsf{Sim}(1^\lambda, \mathsf{pk}, \mathsf{tr}, C, C(x))\right) \leq \nu(\lambda),$$

  where the randomness is *only* over the randomness of Hint, and all other random values are fixed.

- **Sender's Statistical Indistinguishable Security Against Semi-Malicious Receiver.** For any input string x, any circuit $C : \{0,1\}^{|x|} \to \{0,1\}$ with $C(x) = 1$, any unbounded adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that

$$\left|\Pr\left[(r, r') \leftarrow \mathcal{A}(1^\lambda), \substack{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda; r) \\ (\mathsf{pk},\mathsf{msk}) = \mathsf{KeyGen}(\mathsf{crs}, x; r')} : \mathcal{A}(\mathsf{Enc}(\mathsf{pk}, C, 0)) = 1\right] - \right.$$
$$\left.\Pr\left[(r, r') \leftarrow \mathcal{A}(1^\lambda), \substack{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda; r), \\ (\mathsf{pk},\mathsf{msk}) = \mathsf{KeyGen}(\mathsf{crs}, x; r')} : \mathcal{A}(\mathsf{Enc}(\mathsf{pk}, C, 1)) = 1\right]\right| \leq \nu(\lambda)$$

2AB-SFE. A 2AB-SFE scheme with public decryption has the same syntax as AB-SFE with public decryption, except that Enc and Dec are replaced by the following two algorithms:

- $\mathsf{2Enc}(\mathsf{pk}, C, \{\mu_{i,0}, \mu_{i,1}\}_{i\in[\ell_{\text{out}}]})$: On input the public key pk, a multi-bit output circuit $C : \{0,1\}^{|x|} \to \{0,1\}^{\ell_{\text{out}}}$, and ℓ_{out} pair of labels, it output a ciphertext ct.
- $\mathsf{2Dec}(\mathsf{sk}_C, \mathsf{ct})$: On input a hint sk_C, and a ciphertext ct, output $\{\mu'_i\}_{i\in[\ell_{\text{out}}]}$.

We also extend the correctness and sender's security to the following.

- **Correctness:** For any binary string x, circuit $C : \{0,1\}^{|x|} \to \{0,1\}^{\ell_{\text{out}}}$, and any messages $(\mu_{i,0}, \mu_{i,1})_{i\in[\ell_{\text{out}}]}$, there exists a negligible function $\nu(\lambda)$ such that

$$\Pr\left[\substack{\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),(\mathsf{pk},\mathsf{msk})\leftarrow\mathsf{KeyGen}(\mathsf{crs},x),\mathsf{ct}\leftarrow\mathsf{2Enc}(\mathsf{pk},C,(\mu_{i,0},\mu_{i,1})_{i\in[\ell_{\text{out}}]})\\ \mathsf{sk}_C\leftarrow\mathsf{Hint}(\mathsf{msk},C),(\mu'_i)_{i\in[\ell_{\text{out}}]}\leftarrow\mathsf{2Dec}(\mathsf{sk}_C,\mathsf{ct})} : \forall i \in [\ell_{\text{out}}], \mu'_i = \mu_{i,C_i(x)}\right] \geq 1 - \nu(\lambda).$$

  where $C_i(x)$ is the i-th output bit of $C(x)$.

- **Sender's Statistical Indistinguishable Security Against Semi-Malicious Receiver:** For any unbounded adversary $\mathcal{A}$, any binary string x, any circuit $C : \{0,1\}^{|x|} \to \{0,1\}^{\ell_{\text{out}}}$, and any two sets of labels $\{\mu_{i,0}\}_{i\in[\ell_{\text{out}}]}, \{\mu_{i,1}\}_{i\in[\ell_{\text{out}}]}$, there exists a negligible function $\nu(\lambda)$ such that

$$\Bigg|\Pr\left[(r,r')\leftarrow\mathcal{A}(1^\lambda), \substack{\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda;r)\\ (\mathsf{pk},\mathsf{msk})=\mathsf{KeyGen}(\mathsf{crs},x;r')}, \mathcal{A}(\mathsf{2Enc}(\mathsf{pk}, C, (\mu_{i,0}, \mu_{i,1})_{i\in[\ell_{\text{out}}]})) = 1\right] -$$
$$\Pr\left[(r,r')\leftarrow\mathcal{A}(1^\lambda), \substack{\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda;r)\\ (\mathsf{pk},\mathsf{msk})=\mathsf{KeyGen}(\mathsf{crs},x;r')} : \mathcal{A}(\mathsf{2Enc}(\mathsf{pk}, C, (\mu_{i,C_i(x)}, \mu_{i,C_i(x)})_{i\in[\ell_{\text{out}}]})) = 1\right]\Bigg| \leq \nu(\lambda)$$

From AB-SFE to 2AB-SFE with Public Decryption. Given an AB-SFE scheme with public decryption, it is straightforward to construct a 2AB-SFE scheme with public decryption, following the methodology in [22] (where it was described in the context of attribute-based encryption). Roughly speaking, the idea is to encrypt one of the messages under the *complement* of C. We refer the reader to [22] for details.

4.2 Construction

Our construction uses the following parameters and algorithms.

- λ, the security parameter.
- n, the dimension of LWE.
- $q = 2^{\Theta(d \log^3 \lambda)}$, the LWE modulus, where d is the bound for the depth of the circuit. We choose q to be a prime.
- D_σ, the discrete Gaussian of deviation $\sigma = \mathsf{poly}(\lambda)$. We assume that when the randomness are chosen maliciously, we have $|x| < \lambda \cdot \sigma$ for any $x \leftarrow D_\sigma$.
- $\sigma_2 = \mathsf{poly}(\lambda), \sigma_1 = \sigma_2 \cdot \sigma \cdot 2^{\log^2 \lambda}, \sigma_3 = \sigma_2 \cdot \sigma \cdot 2^{\Theta(d \log^2 \lambda)}$ are the deviations for discrete Gaussians.
- $D_{\sigma'}$, the discrete Gaussian of deviation $\sigma' = 2^{\Theta(d \log^2 \lambda)}$. We also assume that when the randomness are chosen maliciously, we have $|x| < \lambda \cdot \sigma'$, for any $x \leftarrow D_{\sigma'}$.

- $m = \Omega(n \log q)$, the number of columns in the commitment.
- A homomorphic commitment scheme (Gen, Com, Eval). See Section 3.5.
- Basis sampling algorithm SampleBasis, with deviation $\sigma' = 2^{\Theta(d \log^2 m)}$. See Theorem 3.5.
- $\mathsf{Ext}(X, \mathbf{r})$: a seeded strong extractor in Theorem 3.1, where X is a random variable with sufficient min-entropy, and the seed $\mathbf{r} \leftarrow \{0,1\}^\lambda$ is uniformly at random.

The construction is described below.

- Setup(1^λ):
  - Sample $\mathbf{B} \leftarrow \mathbb{Z}_q^{n/2 \times m}, \mathbf{S} \leftarrow D_\sigma^{n/2 \times n/2}, \mathbf{E} \leftarrow D_\sigma^{n/2 \times m}$, and let $\mathbf{A} = \begin{bmatrix} \mathbf{B} \\ \mathbf{S} \cdot \mathbf{B} + \mathbf{E} \end{bmatrix}$.
  - Sample $\widehat{\mathbf{B}} \leftarrow \mathbb{Z}_q^{n/2 \times 2m}, \widehat{\mathbf{E}} \leftarrow D_\sigma^{n/2 \times 2m}$, let $\widehat{\mathbf{A}} = \begin{bmatrix} \widehat{\mathbf{B}} \\ \mathbf{S} \cdot \widehat{\mathbf{B}} + \widehat{\mathbf{E}} \end{bmatrix}$.
  - Output crs $= (\mathbf{A}, \widehat{\mathbf{A}})$.

- KeyGen(crs, $x = (x_1, \ldots, x_u) \in \{0,1\}^u$):
  - Parse crs $= (\mathbf{A}, \widehat{\mathbf{A}})$.
  - For all $i \in [u]$, sample $\mathbf{R}_i \leftarrow \{0,1\}^{m \times m}$, and let $\mathbf{C}_i = \mathsf{Com}(\mathbf{A}, x_i; \mathbf{R}_i) = \mathbf{A} \cdot \mathbf{R}_i + x_i \mathbf{G}$.
  - Let pk $= (\text{crs}, \{\mathbf{C}_i\}_{i \in [u]})$, and msk $= (\text{pk}, \{\mathbf{R}_i\}_{i \in [u]})$.
  - Output (pk, msk).

- Enc(pk, $C, \mu \in \{0,1\}$):
  - Parse pk $= (\text{crs}, \{\mathbf{C}_i\}_{i \in [u]})$ and crs $= (\mathbf{A}, \widehat{\mathbf{A}})$.
  - Deterministically homomorphically evaluate $\mathbf{C}_C = \mathsf{Eval}(C, \{\mathbf{C}_i\}_{i \in [u]})$.
  - Sample $\mathbf{z}_1 \leftarrow D_{\sigma_1}^{1 \times n/2}, \mathbf{z}_2 \leftarrow D_{\sigma_2}^{1 \times n/2}$ and $\mathbf{e} \leftarrow D_{\sigma_3}^{1 \times 4m}, r \leftarrow \{0,1\}^\lambda$.
  - Let $\mathbf{A}' = \begin{bmatrix} \mathbf{A} & \mathbf{C}_C \end{bmatrix}$, and $\mathbf{A}'' = \begin{bmatrix} \widehat{\mathbf{A}} & \mathbf{A}' \end{bmatrix}$.
  - Output ct $= \left(\begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \cdot \mathbf{A}'' + \mathbf{e}, \mathsf{Ext}(\mathbf{z}, \mathbf{r}) \oplus \mu, \mathbf{r}\right)$.

- Hint(msk, C):
  - Parse msk $= (\text{pk}, \{\mathbf{R}_i\}_{i \in [u]})$, pk $= (\text{crs}, \{\mathbf{C}_i\}_{i \in [u]})$, and crs $= (\mathbf{A}, \widehat{\mathbf{A}})$.
  - Deterministically homomorphically evaluate $\mathbf{C}_C = \mathsf{Eval}(C, \{\mathbf{C}_i\}_{i \in [u]})$.
  - If $C(x) = 0$, Let $\text{sk}_C = \perp$. Otherwise, parse $\mathbf{C}_C = \mathbf{A} \cdot \mathbf{R}_{C,x} + \mathbf{G}$, where $\mathbf{R}_{C,x}$ can be computed from $\{\mathbf{R}_i\}_{i \in [u]}$ and C in polynomial time.
  - Let $\mathbf{A}' = \begin{bmatrix} \mathbf{A} & \mathbf{C}_C \end{bmatrix}$, and $\mathbf{A}'' = \begin{bmatrix} \widehat{\mathbf{A}} & \mathbf{A}' \end{bmatrix}$.

- Let $\mathbf{T}_{\mathbf{A}'} = \begin{bmatrix} \mathbf{I} & -\mathbf{R}_{C,x} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{G}^{-1}[\mathbf{A}] & \mathbf{T}_\mathbf{G} \end{bmatrix}$, where $\mathbf{T}_\mathbf{G}$ is the short basis for $\Lambda_q^\perp(\mathbf{G})$.
- Sample $\mathbf{H} \leftarrow \mathsf{SampleBasis}(\mathbf{A}'', \mathbf{T}_{\mathbf{A}'}, 2, \sigma')$.
- Output $\mathsf{sk}_C = (\mathbf{H}, \mathbf{A}'')$.

- $\underline{\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct})\text{:}}$
  - If $\mathsf{sk}_C = \perp$, output $\perp$.
  - Otherwise, parse $\mathsf{sk}_C = (\mathbf{H}, \mathbf{A}'')$, and $\mathsf{ct} = (\mathbf{C}, c, \mathbf{r})$.
  - Compute $\mathbf{C}' = \mathbf{C} \cdot \mathbf{H} \pmod q$. Represent each element of $\mathbf{C}'$ in $[-q/2, q/2)$, and solve over $\mathbb{Z}$ the linear equation $\mathbf{x} \cdot \mathbf{H} = \mathbf{C}'$ about $\mathbf{x}$.
  - Solve over $\mathbb{Z}_q$ the linear equation $\mathbf{z}' \cdot \widehat{\mathbf{A}}'' = \mathbf{C} - \mathbf{x}$ about $\mathbf{z}'$.
  - Output $m' = c \oplus \mathsf{Ext}(\mathbf{z}, \mathbf{r})$.

Now we proceed to prove the properties.

Lemma 4.1 (Correctness). *The above construction satisfies correctness.*

Proof. For any binary string x, any circuit C with $C(x) = 1$ and depth at most d, and any message $\mu \in \{0,1\}$, by Lemma 3.6, $\mathbf{R}_{C,x}$ is bounded by $2^{O(d \log m)}$. Hence, $\|\mathbf{T}_{\mathbf{A}'}\|_2 \le (2 + \|\mathbf{R}_{C,x}\|_2) \cdot O(m) = 2^{O(d \log m)}$, and thus $\|\widetilde{\mathbf{T}_{\mathbf{A}'}}\| \le \|\mathbf{T}_{\mathbf{A}'}\|_2 = 2^{O(d \log m)}$. Since the matrix $\mathbf{T}_{\mathbf{A}'}$ is basis for $\Lambda_q^\perp(\mathbf{A}')$ and we set the parameter $\sigma' = 2^{\Theta(d \log^2 m)} > \|\widetilde{\mathbf{T}_{\mathbf{A}'}}\| \cdot \sqrt{2m} \cdot \omega(\sqrt{\log 2m})$. From Theorem 3.5, $\mathbf{H}$ is a basis for $\Lambda_q^\perp(\mathbf{A}'')$, where $\mathbf{A}'' = \begin{bmatrix} \widehat{\mathbf{A}} & \mathbf{A}' \end{bmatrix}$. Hence, we have $\mathbf{C}' = \mathbf{C} \cdot \mathbf{H} \pmod q = \mathbf{e} \cdot \mathbf{H} \pmod q$. Since $\mathbf{e}$ and $\mathbf{H}$ are both small matrices, and $\|\mathbf{e}\mathbf{H}\|_\infty \le \|\mathbf{e}\|_\infty \cdot \|\mathbf{H}\|_\infty \cdot O(m) < q$, we know that $\mathbf{e} \cdot \mathbf{H} = \mathbf{C}'$. From the fact that $\mathbf{H}$ is a basis, we know $\mathbf{H}$ is full rank over $\mathbb{R}^{4m}$. Hence, the (unique) solution $\mathbf{x}$ to the linear equation $\mathbf{x} \cdot \mathbf{H} = \mathbf{C}'$ is $\mathbf{e}$. Since $\mathbf{A}'' = \begin{bmatrix} \widehat{\mathbf{A}} & \mathbf{A}' \end{bmatrix}$ is full rank over $\mathbb{Z}_q$ with overwhleming probability, the (unique) solution $\mathbf{z}'$ of the linear equation $\mathbf{z}' \cdot \mathbf{A}'' = \mathbf{C} - \mathbf{x}$ is $\mathbf{z}$. Hence, $m' = c \oplus \mathsf{Ext}(\mathbf{z}, \mathbf{r}) = \mu$, and we prove the correctness. □

Lemma 4.2 (Statistical Indistinguishability of Public Keys). *The construction satisfies statistical public key indistinguishability security.*

Proof. For any x_0, x_1 with $|x_0| = |x_1|$, and $b \in \{0,1\}$, we have $\mathsf{pk}_b = (\mathsf{crs}, \mathsf{Com}(\mathbf{A}, x_b))$. From the statistical hiding property of the commitment scheme, we have that $\mathsf{SD}(\mathsf{pk}_0, \mathsf{pk}_1)$ is negligible. □

Lemma 4.3 (Simulation of Hints). *The construction satisfies hint simulation security.*

Proof. We construct the following simulator $(\overline{\mathsf{Setup}}, \mathsf{Sim})$. □

We now prove the two properties. For any $x \in \{0,1\}^n$, let $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, x)$, and $(\overline{\mathsf{crs}}, \mathsf{tr}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$.

- crs **and** $\overline{\mathsf{crs}}$ **are Computationally Indistinguishable.** This follows from the property that $\widehat{\mathbf{A}}$ sampled by $\mathsf{TrapGen}$ is statistically close to uniform random, and hence $\overline{\mathsf{crs}}$ is statistically close to the uniform distribution. On the other hand, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ is also computationally indistinguishable from uniform random matrices by LWE assumption.

Simulator $(\overline{\mathsf{Setup}}, \mathsf{Sim})$

- $\overline{\mathsf{Setup}}(1^\lambda)$:
  - Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$.
  - Let $(\widehat{\mathbf{A}}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, 1^{2m})$.
  - Output $\overline{\mathsf{crs}} = (\mathbf{A}, \widehat{\mathbf{A}})$, and trapdoor $\mathsf{tr} = \mathbf{T}$.

- $\mathsf{Sim}(1^\lambda, \mathsf{pk}, \mathsf{tr}, C, C(x))$:
  - Parse $\mathsf{pk} = (\overline{\mathsf{crs}}, (\mathbf{C}_i)_{i\in[u]})$, $\overline{\mathsf{crs}} = (\mathbf{y}, \mathbf{A}, \widehat{\mathbf{A}})$, and $\mathsf{tr} = \mathbf{T}$.
  - Deterministically homomorphically compute $\mathbf{C}_C = \mathsf{Eval}(C, (\mathbf{C}_i)_{i\in[u]})$.
  - Denote $\mathbf{A}' = \begin{bmatrix}\mathbf{A} & \mathbf{C}_C\end{bmatrix}$, and $\mathbf{A}'' = \begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}'\end{bmatrix}$.
  - If $C(x) = 0$, Let $\overline{\mathsf{sk}_C} = \perp$.
  - Otherwise, Sample $\mathbf{H} \leftarrow \mathsf{SampleBasis}(\mathbf{A}'', \mathbf{T}, 1, \sigma')$.
  - Output $\overline{\mathsf{sk}_C} = (\mathbf{H}, \mathbf{A}'')$.

Figure 1: Description of the simulator.

- $\mathsf{SD}\left(\mathsf{Hint}(\mathsf{msk}, C), \mathsf{Sim}(1^\lambda, \mathsf{pk}, \mathsf{tr}, C, C(x))\right) \le \mathsf{negl}(\lambda)$: Note that the matrices $\mathbf{A}'' = \begin{bmatrix}\widehat{\mathbf{A}} & \mathbf{A}'\end{bmatrix}$ in Sim and Hint are the same. Follow the same argument in Lemma 4.1, the parameters σ' is large enough for the condition in Theorem 3.5. Hence, from Theorem 3.5, the distribution of the basis $\mathbf{H}$ only depends on $\mathbf{A}''$ and σ'. Thus we have

$$\mathsf{SD}\left(\mathsf{SampleBasis}(\mathbf{A}'', \mathbf{T}_{\mathbf{A}'}, 2, \sigma'), \mathsf{SampleBasis}(\mathbf{A}'', \mathbf{T}, 1, \sigma')\right) \le \mathsf{negl}(\lambda).$$

  Hence, $\mathsf{SD}(\mathsf{sk}_C, \overline{\mathsf{sk}_C}) \le \mathsf{negl}(\lambda)$.

Lemma 4.4 (Sender's Statistical Indistinguishability Security Against Semi-Malicious Receiver)**.** *The above construction achieves sender's statistical indistinguishability security against semi-malicious receiver.*

Proof. For any input $x = (x_1, \ldots, x_u)$ and circuit C with $C(x) = 0$, and any unbounded adversary $\mathcal{A}$, we build the following hybrids. Note that since the adversary is semi-malicious, in the following hyrbids, the matrix $\mathbf{A} = \begin{bmatrix}\mathbf{B}^* \\ \mathbf{S}^*\cdot\mathbf{B}^* + \mathbf{E}^*\end{bmatrix}$, and $\widehat{\mathbf{A}} = \begin{bmatrix}\widehat{\mathbf{B}}^* \\ \mathbf{S}^*\cdot\widehat{\mathbf{B}}^* + \widehat{\mathbf{E}}^*\end{bmatrix}$, where $\mathbf{S}^*, \mathbf{E}^*, \widehat{\mathbf{E}}^*$ are sampled with maliciously chosen randomness, but their $\|\cdot\|$ norms are still bounded by $\|\mathbf{S}^*\| \le \lambda\cdot\sigma_1\cdot O(\sqrt{m})$.

- Hybrid_0: In this hybrid, the adversary is given a ciphertext of $\mathsf{Enc}(\mathsf{pk}, C, 0)$.

- Hybrid_1: Let $\mathbf{R}_{C,x}$ be the small matrix such that $\mathbf{C}_C = \mathbf{A}\cdot\mathbf{R}_{C,x}$. Then the first coordinate of ct in

Hybrid$_0$ is

$$\begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \cdot \begin{bmatrix} \widehat{\mathbf{A}} & \mathbf{A} & \mathbf{A}\mathbf{R}_{C,x} \end{bmatrix} + \mathbf{e} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \cdot \begin{bmatrix} \widehat{\mathbf{B}}^* & \mathbf{B}^* & \mathbf{B}^*\mathbf{R}_{C,x} \\ \mathbf{S}^*\widehat{\mathbf{B}}^* + \widehat{\mathbf{E}} & \mathbf{S}^*\mathbf{B}^* + \mathbf{E}^* & \mathbf{S}^*\mathbf{B}^*\mathbf{R}_{C,x} + \mathbf{E}^*\mathbf{R}_{C,x} \end{bmatrix} + \mathbf{e} \quad (1)$$

$$= \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}^* & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \widehat{\mathbf{B}}^* & \mathbf{B}^* & \mathbf{B}^*\mathbf{R}_{C,x} \\ \widehat{\mathbf{E}} & \mathbf{E}^* & \mathbf{E}^*\mathbf{R}_{C,x} \end{bmatrix} + \mathbf{e} \quad (2)$$

$$= \begin{bmatrix} \mathbf{z}_1 + \mathbf{z}_2\mathbf{S}^* & \mathbf{z}_2 \end{bmatrix} \cdot \begin{bmatrix} \widehat{\mathbf{B}}^* & \mathbf{B}^* & \mathbf{B}^*\mathbf{R}_{C,x} \\ \widehat{\mathbf{E}} & \mathbf{E}^* & \mathbf{E}^*\mathbf{R}_{C,x} \end{bmatrix} + \mathbf{e} \quad (3)$$

This hybrid is almost the same as Hybrid$_0$, except that we replace the first coordinate of ct as Equation (3). From the above argument, Hybrid$_0$ and Hybrid$_1$ are identical.

- Hybrid$_2$: This hybrid is the same as Hybrid$_1$, except that we further replace the first coordinate of ct as follows.

$$\begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \cdot \begin{bmatrix} \widehat{\mathbf{B}}^* & \mathbf{B}^* & \mathbf{B}^*\mathbf{R}_{C,x} \\ \widehat{\mathbf{E}} & \mathbf{E}^* & \mathbf{E}^*\mathbf{R}_{C,x} \end{bmatrix} + \mathbf{e}$$

  Hybrid$_1$ and Hybrid$_2$ are statistically indisitnguishable, because we set the parameter $\sigma_1 = \sigma \cdot \sigma_2 \cdot 2^{\log^2 \lambda}$ large enough such that $\|\mathbf{z}_1 + \mathbf{z}_2\mathbf{S}^*\|/\sigma_1 = \mathsf{negl}(\lambda)$. By noise flooding (Theorem 3.2), $\mathbf{z}_1 + \mathbf{z}_2\mathbf{S}^*$ is statistically close to $\mathbf{z}_1$.

- Hybrid$_3$: This hyrbid is almost the same as Hybrid$_2$, except that we replace the first coordinate of ct as follows.

$$\begin{bmatrix} \mathbf{z}_1 \cdot \widehat{\mathbf{B}}^* + \mathbf{e}_1, & \mathbf{z}_1 \cdot \mathbf{B}^* + \mathbf{e}_2, & \mathbf{z}_1 \cdot \mathbf{B}^*\mathbf{R}_{C,x} + \mathbf{e}_3 \end{bmatrix}$$

  where $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) = \mathbf{e}$.

  We will argue that Hybrid$_2$ and Hybrid$_3$ are statistically indistinguishable. Since the term $\mathbf{z}_2 \cdot \widehat{\mathbf{E}}, \mathbf{z}_2 \cdot \mathbf{E}^*, \mathbf{z}_2 \cdot \mathbf{E}^*\mathbf{R}_{C,x}$ in Hybrid$_2$ are small matrices, and we set the deviation σ_3 for $\mathbf{e}$ large enough. Hence, by noise flooding (Theorem 3.2), $\begin{bmatrix} \mathbf{z}_2 \cdot \widehat{\mathbf{E}} & \mathbf{z}_2 \cdot \mathbf{E}^* & \mathbf{z}_2 \cdot \mathbf{E}^*\mathbf{R}_{C,x} \end{bmatrix} + \mathbf{e}$ and $\mathbf{e}$ are statistically close. Therefore, Hybrid$_2$ and Hybrid$_3$ are statistically indistinguishable.

- Hybrid$_4$: This hyrbid is almost the same as Hybrid$_3$, except that the second coordinate of ct is replaced with a uniformly random bit.

  Hybrid$_2$ and Hybrid$_4$ are also statistically indistinguishable, since the entropy of $\mathbf{z}$ conditioned on the first term of ct is at least $\mathsf{H}_\infty(\mathbf{z}_2) \geq n/2$. Since Ext is a strong extractor, by extraction, the second and the third term $(\mathsf{Ext}(\mathbf{z}, \mathbf{r}) \oplus \mu, \mathbf{r})$ is statistically close to $(b, \mathbf{r})$, where $b \leftarrow \{0, 1\}$ is uniformly random.

- Hybrid$_5$: This hybrid is almost the same as Hybrid$_0$, except that the adversary is given a ciphertext of $\mathsf{Enc}(\mathsf{pk}, C, 1)$.

  Hybrid$_4$ and Hybrid$_5$ are also statistically indistinguishable, since Hybrid$_4$ does not depends on μ, then the same argument from Hybrid$_0$ to Hybrid$_4$ can be applied from Hybrid$_4$ to Hybrid$_5$.

By the hyrbid argument, we finish the proof. □

Removing the Depth Dependence. In the above construction, the parameters depends on the depth of the circuit. However, one can use the randomized encoding [4] to remove the depth dependence. Specifically, instead of evaluating the circuit C on input x directly, we evaluate the randomized encoding of C and x. Since the randomized encoding can be computed in NC^1, we can set the parameters to be large enough to work for any circuit in NC^1, and thus remove the depth dependence.

5 Unbounded MPC

5.1 Definition

A (semi-malicious) unbounded MPC protocol is a 2-round MPC protocol $(\mathsf{Round}_1, \mathsf{Round}_2, \mathsf{Rec})$ satisfying the following syntax.

- **First Round:** The i-th party's input is x_i. It sets the random coins r_i, and executes $\mathsf{msg}_i \leftarrow \mathsf{Round}_1(1^\lambda, x_i; r_i)$. Then the i-th party broadcasts msg_i.
- **Second Round:** After receiving the first round messages, a subset of parties $S \subseteq [N]$ decide to jointly compute a ℓ_{out}-bit output circuit $f : \prod_{i \in S}\{0,1\}^{|x_i|} \to \{0,1\}^{\ell_{\mathsf{out}}}$.

  For each $i \in S$, the i-th party executes $p_i \leftarrow \mathsf{Round}_2(x_i, r_i, \{\mathsf{msg}_j\}_{j\in S}, S, f)$, and broadcasts p_i.
- **Public Recovery:** Anyone with $\{p_i\}_{i \in S}$ can execute $y \leftarrow \mathsf{Rec}(\{p_i\}_{i\in S}, S)$.

Efficiency. The running time of Round_1 is polynomial in λ and $|x_i|$, and is independent of N or the size of the circuit they want to compute later. The running time of Round_2 is polynomial in λ, $|S|$ and $|C|$.

Unbounded-Party Semi-Malicious Security. For any PPT adversary $\mathcal{A}$, there exists a simulator $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that

$$\left|\Pr\left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\overline{\mathsf{Regstr}}(\cdot,\cdot),\overline{\mathsf{Eval}}(\cdot,\cdot)}(1^\lambda) = 1\right]\right| \le \mathsf{negl}(\lambda)$$

where the oracles $\mathsf{Regstr}(\cdot,\cdot)$ and $\mathsf{Eval}(\cdot,\cdot)$ are defined as follows.

- $\mathsf{Regstr}(\mathsf{flag} \in \{\mathsf{Honest}, \mathsf{Dishonest}\}, (x, r^*))$**:** This oracle allows the adversary to register a new party. It returns the first round message msg_N of the new party (indexed by N). When a new party is registered, we allow the adversary to choose the input x and whether the new party is corrupted or not (specified by flag). If the adversary decides to corrupt it, then we also allow the adversary to choose the random coins r^*. A more specific description is as follows. We use a set H to maintain the set of honest parties.
  - If flag is Honest, then
    - Let $H = H \cup \{i\}$
    - Set random coins r_N, and let $\mathsf{msg}_i = \mathsf{Round}_1(1^\lambda, x; r_N)$.
    - Output msg_N.
  - Otherwise, if flag is $\mathsf{Dishonest}$, then let $\mathsf{msg}_N = \mathsf{Round}_1(1^\lambda, x; r^*)$.
  - Let $x_N = x$ and $N = N + 1$.

- $\mathsf{Eval}(S, f)$**:** This oracle allows the adversary to query a function f, which only acts on a subset of parties $S \subseteq [N]$. It returns the second round messages of the honest parties.
  - If $S \nsubseteq [N]$, then abort.
  - For each $i \in S \cap H$, let $p_i \leftarrow \mathsf{Round}_2(x_i, r_i, \{\mathsf{msg}_j\}_{j \in S}, S, f)$.
  - Output $\{p_i\}_{i \in S \cap H}$.
- $\overline{\mathsf{Regstr}}(\mathsf{flag} \in \{\mathsf{Honest}, \mathsf{Dishonest}\}, (x, r^*))$**:** This oracle simulates the oracle Regstr using the simulator Sim_1. When the adversary registers an honest party, then it invokes the simulator Sim_1 to generate the first round message. Otherwise, it runs the honest protocol Round_1 to generate first round messages for the corrupted parties.
  - If flag is Honest, then
    - Let $H = H \cup \{N\}$.
    - Let $(\mathsf{msg}_N, \mathsf{st}_N) \leftarrow \mathsf{Sim}_1(1^\lambda, 1^{|x|})$.
    - Output msg_N.
  - Otherwise, $\mathsf{Round}_1(1^\lambda, x; r_N)$.
  - Let $x_N = x$, and $N = N + 1$.
- $\overline{\mathsf{Eval}}(S, f)$**:** This oracle simulates the oracle Eval using the simulator Sim_2. It returns the simulated second round messages for all the honest parties in S.
  - If $S \nsubseteq [N]$, then abort.
  - Output $\{p_i\}_{i \in S \cap H} \leftarrow \mathsf{Sim}_2(\{\mathsf{st}_i\}_{i \in S \cap H}, S, H, f, f(\{x_i\}_{i \in S \cap H}))$.

5.2 Construction

We present our unbounded MPC protocol $\Pi = (\mathsf{Round}_1, \mathsf{Round}_2, \mathsf{Rec})$ in Figure 2. For an overview of the construction, see Section 2.3. Our construction uses the following ingredients:

- An AB-SFE scheme ABSFE = (ABSFE.Setup, ABSFE.KGen, ABSFE.2Enc, ABSFE.Hint, ABSFE.2Dec) with public decryption.
- A *one-time use* two-round semi-malicious MPC protocol One = ($\mathsf{One.Round}_1$, $\mathsf{One.Round}_2$, One.Rec) in the plain model.
- A pseudorandom function PRF = (PRF.Gen, PRF.Eval).
- A garbling scheme GC = (GC.Garble, GC.Eval).

5.3 Security

Lemma 5.1 (Unbounded-Party Semi-Malicious Simulation Security)**.** *The construction in Section 5.2 satisfies semi-malicious unbounded-party simulation security.*

$\underline{\mathsf{Round}_1(1^\lambda, x_i)}$: Party i performs the following steps:

- Sample a CRS $\mathsf{crs}_i \leftarrow \mathsf{ABSFE.Setup}(1^\lambda)$ and a PRF key $k_i \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Compute $(\mathsf{pk}_i, \mathsf{msk}_i) \leftarrow \mathsf{ABSFE.KGen}(\mathsf{crs}_i, (x_i, k_i))$
- Output $\mathsf{msg}_i = \mathsf{pk}_i$.

$\underline{\mathsf{Round}_2(x_i, r_i, \{\mathsf{msg}_j\}_{j\in S}, S, f)}$: Party i performs the following steps:

- Compute crs_i, k_i and msk_i from r_i, and parse $\mathsf{msg}_j = \mathsf{pk}_j$.
- Compute $(\widetilde{C}_i, \widetilde{\mathsf{lab}}) \leftarrow \mathsf{GC.Garble}(C_{[x_i,k_i]})$, where the circuit $C_{[x_i,k_i]}$ on input a tuple $\{\widetilde{\mathsf{msg}_j}\}_{j\in S}$ does the following:
  - $r_i = \mathsf{PRF.Eval}(k_i, (S \,||\, f))$.
  - Output $\widetilde{p_i} = \mathsf{One.Round}_2(x_i, r_i, \{\widetilde{\mathsf{msg}_j}\}_{j\in S}, f)$.
- Parse $\widetilde{\mathsf{lab}} = \{\widetilde{\mathsf{lab}_{j,k,b}}\}_{j\in S, k\in[|\widetilde{\mathsf{msg}_j}|], b\in\{0,1\}}$.
- For $j \in S \setminus \{i\}$, compute $c_{i,j} \leftarrow \mathsf{ABSFE.2Enc}\left(\mathsf{pk}_j, G_{S,f}, \{\widetilde{\mathsf{lab}_{j,k,0}}, \widetilde{\mathsf{lab}_{j,k,1}}\}_{k\in[|\widetilde{\mathsf{msg}_j}|]}\right)$, where the circuit $G_{S,f}$ on input (x_i, k_i) does the following:
  - $r_i = \mathsf{PRF.Eval}(k_i, (S \,||\, f))$.
  - $\widetilde{\mathsf{msg}_i} = \mathsf{One.Round}_1(1^\lambda, x_i, f; r_i)$.
  - Output $\widetilde{\mathsf{msg}_i}$.
- $h_i \leftarrow \mathsf{ABSFE.Hint}(\mathsf{msk}_i, G_{S,f})$, $\widetilde{\mathsf{msg}_i} = G_{S,f}(x_i, k_i)$.
- Output $p_i = \left(\{c_{i,j}\}_{j\in S\setminus\{i\}}, h_i, \widetilde{C}_i, \{\widetilde{\mathsf{lab}_{i,k,\widetilde{\mathsf{msg}_i}[k]}}\}_{k\in[|\widetilde{\mathsf{msg}_i}|]}\right)$.

$\underline{\mathsf{Rec}(\{p_j\}_{j\in S}, S)}$: Party i performs the following steps:

- For each $i \in S$, parse $p_i = \left(\{c_{i,j}\}_{j\in S\setminus\{i\}}, h_i, \widetilde{C}_i, \{\widetilde{\mathsf{lab}_{i,k,\widetilde{\mathsf{msg}_i}[k]}}\}_{k\in[|\widetilde{\mathsf{msg}_i}|]}\right)$.
- For each $i \in S$ and $j \in S \setminus \{i\}$, compute $\widetilde{\mathsf{lab}}_{i,j} \leftarrow \mathsf{ABSFE.2Dec}(h_i, c_{i,j})$. Set $\widetilde{\mathsf{lab}}_{i,i} = \{\widetilde{\mathsf{lab}_{i,k,\widetilde{\mathsf{msg}_i}[k]}}\}_{k\in[|\widetilde{\mathsf{msg}_i}|]}$. Compute $\widetilde{p_i} = \mathsf{GC.Eval}(\widetilde{C}_i, \{\widetilde{\mathsf{lab}}_j\}_{j\in S})$.
- Output $y = \mathsf{One.Rec}(\{\widetilde{p_i}\}_{i\in S})$.

Figure 2: Description of Unbounded-Party Reusable MPC Π.

Proof. For any n.u. PPT adversary $\mathcal{A}$, let $N(\lambda)$ be the upper bound for the number of parties N that $\mathcal{A}$ registers, and $Q(\lambda)$ be the upper bound for the number of queries that $\mathcal{A}$ makes to Eval. For any

$i^* \in [N(\lambda)]$, and $q^* \in [Q(\lambda)]$, we build the following hybrids.

- Hybrid_0: This hybrid is the same as $\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}$.

- $\mathsf{Hybrid}_1^{(i^*,j^*,q^*)}$: This hybrid is almost the same as the Hybrid_0, except that we replace the labels used by $\mathsf{ABSFE.2Enc}$ to the same labels. Specifically, we replace the $\mathsf{ABSFE.2Enc}$ encryption in $\mathsf{Eval}(\cdot,\cdot)$ as follows.

  - For $j \in S \setminus \{i\}$, if $(i,j,q) < (i^*,j^*,q^*)$, $\widetilde{\mathsf{msg}}_j = G_{S,f}(x_j,k_j)$,
    $c_{i,j} \leftarrow \mathsf{ABSFE.2Enc}(\mathsf{pk}_j, G_{S,f}, \{\widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}}_j[k]}}, \widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}}_j[k]}}\}_{k\in[|\widetilde{\mathsf{msg}}_j|]})$.
    If $(i,j,q) \geq (i^*,j^*,q^*)$, $c_{i,j} \leftarrow \mathsf{ABSFE.2Enc}(\mathsf{pk}_j, G_{S,f}, \{\widetilde{\mathsf{lab}_{j,k,0}}, \widetilde{\mathsf{lab}_{j,k,1}}\}_{k\in[|\widetilde{\mathsf{msg}}_j|]})$.

- $\mathsf{Hybrid}_2^{(i^*,q^*)}$: This hybrid is almost the same as the $\mathsf{Hybrid}_1^{(N,N,Q)+1}$, except that we generate the labels of the garbled circuits by the simulator. Specifically, we replace the garbled circuits generation in $\mathsf{Eval}(\cdot,\cdot)$ as follows.

  - If $(i,q) < (i^*,q^*)$, then $(\overline{C_i}, \overline{\mathsf{lab}}) \leftarrow \mathsf{GC.Sim}(1^\lambda, C_{[x_i,k_i]}(\{\widetilde{\mathsf{msg}}_j\}_{j\in S}))$,
    let $\widetilde{C}_i = \overline{C_i}$, and parse $\overline{\mathsf{lab}} = \{\widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}}_j[k]}}\}_{j\in S, k\in[|\widetilde{\mathsf{msg}}_j|]}$.
    If $(i,q) \geq (i^*,q^*)$, then $(\widetilde{C}_i, \widetilde{\mathsf{lab}}) \leftarrow \mathsf{Garble}(C_{[x_i,k_i]})$.

- $\mathsf{Hybrid}_3^{i^*}$: This hybrid is almost the same as $\mathsf{Hybrid}_2^{(N,Q)+1}$, except that we generate the replace the CRS generation of $\mathsf{Round}_1(1^\lambda, x_i)$ in $\mathsf{Regstr}(\cdot,\cdot)$ as follows.

  - If $i < i^*$ and $i \in H$, generate $(\mathsf{crs}_i, \mathsf{tr}_i) \leftarrow \mathsf{ABSFE}.\overline{\mathsf{Setup}}(1^\lambda)$.
  - If $i \geq i^*$ or $i \notin H$, generate $\mathsf{crs}_i \leftarrow \mathsf{ABSFE.Setup}(1^\lambda)$.

- $\mathsf{Hybrid}_4^{(i^*,q^*)}$: This hybrid is almost the same as Hybrid_3^{N+1}, except that we replace the hint generation in $\mathsf{Eval}(\cdot,\cdot)$ by the simulator. Specifically, let q be the number of queries to $\mathsf{Eval}(\cdot,\cdot)$, we replace the generation of h_i as follows.

  - If $(i,q) < (i^*,q^*)$ and $i \in H$, $h_i \leftarrow \mathsf{ABSFE.Sim}(1^\lambda, \mathsf{pk}_i, \mathsf{tr}_i, G_{S,f}, \widetilde{\mathsf{msg}}_i)$.
  - If $(i,q) \geq (i^*,q^*)$ or $i \in \bar{H}$, $h_i \leftarrow \mathsf{ABSFE.Hint}(\mathsf{msk}_i, G_{S,f})$.

- $\mathsf{Hybrid}_5^{i^*}$: This hybrid is almost the same with $\mathsf{Hybrid}_4^{(N,Q)+1}$, except that we replace the PRF with a random function. Specifically, we replace the randomness r_i generation in $\mathsf{Eval}(\cdot,\cdot)$ with the following. Let (S,f) be the q-th query,

  - For each $i \in S$, if $i < i^*$ and $i \in H$, let $r_i = \mathsf{PRF}_i.\mathsf{F}(S \,||\, f)$.
  - If $i \geq i^*$ or $i \notin H$, let $r_i = \mathsf{PRF.Eval}(k_i, (S \,||\, f))$.
  - Let $\widetilde{\mathsf{msg}}_i = \mathsf{One.Round}_1\left(1^\lambda, x_i, f; r_i\right)$, $\widetilde{p}_i = \mathsf{One.Round}_2(x_i, r_i, \{\widetilde{\mathsf{msg}}_j\}_{j\in S}, f)$.

where $\mathsf{PRF}_i.\mathsf{F}$ is a random function for each $i < i^*, i \in H$.

- $\mathsf{Hybrid}_6^{q^*}$: This hybrid is almost the same with Hybrid_5^{N+1}, except that we replace the $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S\cap H}$ using $\mathsf{One.Sim}$. Specifically, we replace the generation of $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S\cap H}$ in $\mathsf{Eval}(\cdot,\cdot)$ as follows.

  At the begining of $\mathsf{Eval}(\cdot,\cdot)$, we initialize an empty map $\mathsf{Map} : \phi \to \phi$.

  Let (S, f) be the q-th query to $\mathsf{Eval}(\cdot,\cdot)$.

  – If $\mathsf{Map}(S, f)$ is defined before, let $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S\cap H} = \mathsf{Map}(S, f)$.

  – Otherwise, if $q < q^*$, let $r_i = \mathsf{PRF.Eval}(k_i, (S\|f))$ for each $i \in S \setminus H$,

  – $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S\cap H} \leftarrow \mathsf{One.Sim}(1^\lambda, S \cap H, \{x_i, r_i\}_{i \in S\setminus H}, f, f(\{x_i\}_{i\in S}))$,

  – and if $q \geq q^*$, for each $i \in S \cap H$, set fresh randomness r_i,
  let $\widetilde{\mathsf{msg}_i} = \mathsf{One.Round}_1\left(1^\lambda, x_i, f; r_i\right)$, $\widetilde{p_i} = \mathsf{One.Round}_2(x_i, r_i, \{\widetilde{\mathsf{msg}_j}\}_{j\in S}, f)$,
  and define $\mathsf{Map}(S, f) = \{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S\cap H}$.

- Ideal: This hybrid is the same as Hybrid_6^{Q+1}, except that we replace each KGen of real input (x_i, k_i) with the dummy $(0^{|x_i|}, 0^{|k_i|})$, for each $i \in H$. This hybrid is the same as $\mathcal{A}^{\overline{\mathsf{Regstr}}(\cdot,\cdot),\overline{\mathsf{Eval}}(\cdot,\cdot)}(1^\lambda)$. See the simulator in Figure 3.

Lemma 5.2. Hybrid_0 *is identical to* $\mathsf{Hybrid}_1^{(1,1,1)}$*. Moreover, there exists a negligible function $\nu(\lambda)$ such that*

$$\left|\Pr_{\mathsf{Hybrid}_1^{(i^*,j^*,q^*)}}\left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\right] - \Pr_{\mathsf{Hybrid}_1^{(i^*,j^*,q^*)+1}}\left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\right]\right| \leq \nu(\lambda).$$

Proof. We build the following adversary $\mathcal{A}'$ trying to break the sender's indistinguishability security. $\mathcal{A}'$ sets the randomness and runs the adversary $\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}$, where the oracles $\mathsf{Regstr}(\cdot,\cdot)$ and $\mathsf{Eval}(\cdot,\cdot)$ are implemented as follows.

- $\mathsf{Regstr}(\cdot,\cdot)$: For each query, the adversary $\mathcal{A}'$ does the same thing as the Hybrid_0, except that when the adversary $\mathcal{A}$ registers the j^*-th party, if $\mathcal{A}$ decides to corrupt it, then we have the adversary $\mathcal{A}'$ output the random coins $(r, r') = r^*$ chosen by $\mathcal{A}$.

- $\mathsf{Eval}(\cdot,\cdot)$: Let q the q-th query be (S, f). The adversary does the following. For each $i \in H \cap S$, it generates the garbled circuit and labels $(\widetilde{C_i}, \widetilde{\mathsf{lab}})$ for $C_{[x_i,k_i]}$. Then for each $j \in S \setminus \{i\}$, it considers three cases.

  – If $(i, j, q) < (i^*, j^*, q^*)$, $\mathcal{A}'$ uses $\mathsf{ABSFE.2Enc}$ to encrypt the same labels.

  – If $(i, j, q) = (i^*, j^*, q^*)$, it queries the challenger with the circuit $G_{S,f}$, and obtains a challenge ciphertext ct. Let $c_{i,j} = \mathsf{ct}$.

  – If $(i, j, q) > (i^*, j^*, q^*)$, $\mathcal{A}'$ uses $\mathsf{ABSFE.2Enc}$ to encrypt different labels.

  Finally, $\mathcal{A}'$ computes and outputs $\{p_i\}_{i\in S\cap H}$ by the same way as Hybrid_0.

```
$\underline{\mathsf{Sim}_1(1^\lambda, 1^{|x|})\text{:}}$

  – Let $(\overline{\mathsf{crs}_N}, \mathsf{tr}_N) \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$, and $(\mathsf{pk}_N, \mathsf{msk}_N) \leftarrow \mathsf{KGen}(\overline{\mathsf{crs}_N}, (0^{|x|}, 0^\lambda))$.
  – Output $\mathsf{msg}_N = \mathsf{pk}_N$, and $\mathsf{st}_N = \mathsf{tr}_N$.

$\mathsf{Sim}_2$ initialization: an empty map $\mathsf{Map} : \phi \longrightarrow \phi$.

$\underline{\mathsf{Sim}_2(\{\mathsf{st}_i\}_{i \in S \cap H}, S, H, f, f(\{x_i\}_{i \in S \cap H}))\text{:}}$

  – For the $q$-the query $(S, f)$, if $\mathsf{Map}(S, f)$ is defined before, then let
        $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S \cap H} = \mathsf{Map}(S, f)$.
  – Otherwise, let $r_i = \mathsf{PRF.Eval}(k_i, (S||f))$ for each $i \in S \setminus H$, and
        $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S \cap H} \leftarrow \mathsf{One.Sim}(1^\lambda, S \cap H, \{x_i, r_i\}_{i \in S \setminus H}, f, f(\{x_i\}_{i \in S}))$,
    define $\mathsf{Map}(S, f) = \{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i \in S \cap H}$.
  – For each $i \in S \cap H$
      * Let $(\widetilde{C_i}, \widetilde{\mathsf{lab}}) \leftarrow \mathsf{GC.Sim}(1^\lambda, \widetilde{p_i})$, parse $\widetilde{\mathsf{lab}} = \{\widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}_j}[k]}}\}_{j \in S, k \in [|\widetilde{\mathsf{msg}_j}|]}$.
      * For each $j \in S \setminus \{i\}$, compute
            $c_{i,j} \leftarrow \mathsf{ABSFE.2Enc}(\mathsf{pk}_j, G_{S,f}, \{\widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}_j}[k]}}, \widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}_j}[k]}}\}_{k \in [|\widetilde{\mathsf{msg}_j}|]})$.
      * $h_i \leftarrow \mathsf{ABSFE.Sim}(1^\lambda, \mathsf{pk}_i, \mathsf{tr}_i = \mathsf{st}_i, G_{S,f}, \widetilde{\mathsf{msg}_i})$.
  – Output $p_i = \Big((c_{i,j})_{j \in S}, h_i, \widetilde{C_i}, \{\widetilde{\mathsf{lab}_{i,k,\widetilde{\mathsf{msg}_i}[k]}}\}_{k \in [|\widetilde{\mathsf{msg}_i}|]}\Big)$.
```

Figure 3: Description of the simulator $(\mathsf{Sim}_1, \mathsf{Sim}_2)$.

Now for the challenge ciphertext ct, we consider two cases. When ct is obtained by ABSFE.2Enc of different labels, then the adversary $\mathcal{A}'$ simulates the environment of $\mathsf{Hybrid}_1^{(i^*,j^*,q^*)}$. Hence,

$$\Pr\Big[\mathsf{ct} \leftarrow \mathsf{ABSFE.2Enc}(\mathsf{pk}, G_{S,f}, (\widetilde{\mathsf{lab}_{j,k,0}}, \widetilde{\mathsf{lab}_{j,k,1}})_{k \in [|\widetilde{\mathsf{msg}_j}|]}) : \mathcal{A}'(1^\lambda, \mathsf{crs}, r) = 1\Big] = \Pr_{\mathsf{Hybrid}_1^{(i^*,j^*,q^*)}}\Big[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\Big]$$

When ct is generated with the same labels, then the adversary $\mathcal{A}'$ simulates the environment of $\mathsf{Hybrid}_1^{(i^*,j^*,q^*)+1}$. Hence,

$$\Pr\Big[\mathsf{ct} \leftarrow \mathsf{ABSFE.2Enc}(\mathsf{pk}, G_{S,f}, (\widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}_j}[k]}}, \widetilde{\mathsf{lab}_{j,k,\widetilde{\mathsf{msg}_j}[k]}})_{k \in [|\widetilde{\mathsf{msg}_j}|]}) : \mathcal{A}'(1^\lambda, \mathsf{crs}, r) = 1\Big] = \Pr_{\mathsf{Hybrid}_1^{(i^*,j^*,q^*)+1}}\Big[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\Big]$$

From the sender's statistical indistinguishable security of AB-SFE against a semi-malicious receiver, we derive that $\mathsf{Hybrid}_1^{(i^*,j^*,q^*)}$ and $\mathsf{Hybrid}_1^{(i^*,j^*,q^*)+1}$ are indistinguishable. □

Lemma 5.3. *$\mathsf{Hybrid}_1^{(N,N,Q)+1}$ is identical to $\mathsf{Hybrid}_2^{(1,1)}$. Moreover, there exists a negligible function $\nu(\lambda)$ such that*

$$\left| \Pr_{\mathsf{Hybrid}_2^{(i^*,q^*)}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right] - \Pr_{\mathsf{Hybrid}_2^{(i^*,q^*)+1}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right] \right| \leq \nu(\lambda).$$

Proof. We build the following distinguisher $\mathcal{D}$ for the garbled scheme GC. $\mathcal{D}$ takes as input $(1^\lambda, \widetilde{C}, \mathsf{lab})$, sets the randomness and runs the adversary $\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}$, where the oracles $\mathsf{Regstr}(\cdot,\cdot)$ and $\mathsf{Eval}(\cdot,\cdot)$ are implemented as follows.

- $\mathsf{Regstr}(\cdot,\cdot)$: For each query, the adversary $\mathcal{A}'$ does the same thing as the Hybrid_0.
- $\mathsf{Eval}(\cdot,\cdot)$: Let q the q-th query be (S, f). The adversary does the following. For each $i \in H \cap S$, it considers three cases.
  - If $(i,q) < (i^*,q^*)$, then it generates $\widetilde{C}_i, \widetilde{\mathsf{lab}}$ by the simulator GC.Sim.
  - If $(i,q) = (i^*,q^*)$, then it sets $\widetilde{C}_i, \widetilde{\mathsf{lab}}$ to be the input $\widetilde{C}$, lab.
  - If $(i,q) > (i^*,q^*)$, then it generates $\widetilde{C}_i, \widetilde{\mathsf{lab}}$ by honestly garbling $C_{[x_i,k_i]}$.

  Finally, it computes and outputs $\{p_i\}_{i \in S \cap H}$ in the same way as $\mathsf{Hybrid}_1^{(N,N,Q)+1}$.

When $(\widetilde{C}, \mathsf{lab}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C_{[\mathsf{sk}_{i^*},k_{i^*}]})$, then the distinguisher $\mathcal{D}$ simulates the environment of $\mathsf{Hybrid}_2^{(i^*,q^*)}$ for $\mathcal{A}$. Hence, we have

$$\Pr\left[(\widetilde{C}, \mathsf{lab}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C_{[\mathsf{sk}_{i^*},k_{i^*}]}) : \mathcal{D}(1^\lambda, \widetilde{C}, \mathsf{lab}) = 1 \right] = \Pr_{\mathsf{Hybrid}_2^{(i^*,q^*)}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right].$$

When $(\widetilde{C}, \mathsf{lab}) \leftarrow \mathsf{GC.Sim}(1^\lambda, C_{[\mathsf{sk}_{i^*},k_{i^*}]}(\{\widetilde{\mathsf{msg}}_j\}_{j \in S}))$, the distinguisher simulates the environment of $\mathsf{Hybrid}_2^{(i^*,q^*)+1}$ for $\mathcal{A}$. Hence,

$$\Pr\left[(\widetilde{C}, \mathsf{lab}) \leftarrow \mathsf{GC.Sim}(1^\lambda, C_{[\mathsf{sk}_{i^*},k_{i^*}]}(\{\mathsf{msg}_j\}_{j \in S})) : \mathcal{D}(1^\lambda, \widetilde{C}, \mathsf{lab}) = 1 \right] \\ = \Pr_{\mathsf{Hybrid}_2^{(i^*,q^*)+1}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right].$$

From the security of the garbling scheme, we derive that $\mathsf{Hybrid}_2^{(i^*,q^*)}$ and $\mathsf{Hybrid}_2^{(i^*,q^*)+1}$ are indistinguishable. □

Lemma 5.4. *$\mathsf{Hybrid}_2^{(N,Q)+1}$ is identical to Hybrid_3^1. Moreover, there exists a negligible function $\nu(\lambda)$ such that for any n.u. PPT adversary $\mathcal{A}$,*

$$\left| \Pr_{\mathsf{Hybrid}_3^{i^*}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right] - \Pr_{\mathsf{Hybrid}_3^{i^*+1}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right] \right| \leq \nu(\lambda).$$

Proof. We build the following adversary $\mathcal{A}'$ to break the hint simulation property. The adversary $\mathcal{A}'$ takes as input the crs, and for each $i < i^*$, $\mathcal{A}'$ generates the crs_i using $\mathsf{ABSFE}.\overline{\mathsf{Setup}}$. For each $i > i^*$, $\mathcal{A}'$ generates crs_i using $\mathsf{ABSFE.Setup}$. For i^*, if $i^* \in H$, then sets crs_{i^*} as crs. Otherwise, it generates crs_{i^*} using $\mathsf{ABSFE.Setup}$. Then $\mathcal{A}'$ invokes $\mathcal{A}$ and simulates $\mathsf{Regstr}(\cdot,\cdot)$ and $\mathsf{Eval}(\cdot,\cdot)$ in the same way as $\mathsf{Hybrid}_2^{(N,Q)+1}$.

When $\mathsf{crs} \leftarrow \mathsf{ABSFE.Setup}(1^\lambda)$, then $\mathcal{A}'(1^\lambda, \mathsf{crs})$ simulates $\mathsf{Hybrid}_3^{i^*}$ for $\mathcal{A}$. When crs is generated by $\mathsf{ABSFE}.\overline{\mathsf{Setup}}(1^\lambda)$, then $\mathcal{A}'(1^\lambda, \mathsf{crs})$ simulates $\mathsf{Hybrid}_3^{i^*+1}$ for $\mathcal{A}$. From the hint simulation security, we finish the proof. □

Lemma 5.5. Hybrid_3^{N+1} *is identical to* $\mathsf{Hybrid}_4^{(1,1)}$. *Moreover, there exists a negligible function* $\nu(\lambda)$ *such that* $\mathsf{SD}(\mathsf{Hybrid}_4^{(i^*,q^*)}, \mathsf{Hybrid}_4^{(i^*,q^*)+1}) \le \nu(\lambda)$.

Proof. Since the only difference between $\mathsf{Hybrid}_4^{(i^*,q^*)}$ and $\mathsf{Hybrid}_4^{(i^*,q^*)+1}$ is the way that h_i is generated in q-th query of $\mathcal{O}$, from the hint simulation security of AB-SFE, we have $\mathsf{SD}(\mathsf{Hybrid}_4^{(i^*,q^*)}, \mathsf{Hybrid}_4^{(i^*,q^*)+1}) \le \mathsf{negl}(\lambda)$. □

Lemma 5.6. $\mathsf{Hybrid}_4^{(N,Q)+1}$ *and* Hybrid_5^1 *are identical. There exists a negligible function* $\nu(\lambda)$ *such that*

$$\left| \Pr_{\mathsf{Hybrid}_5^{i^*}}\left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\right] - \Pr_{\mathsf{Hybrid}_5^{i^*+1}}\left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\right] \right| \le \nu(\lambda).$$

Proof. We construct the following adversary $\mathcal{A}'$ for the PRF. $\mathcal{A}'^{\mathcal{O}}(1^\lambda)$ is given access to a PRF oracle, and it invokes the adversary $\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda)$ by implementing the oracles $\mathsf{Regstr}(\cdot,\cdot)$ and $\mathsf{Eval}(\cdot,\cdot)$ as follows.

- $\mathsf{Regstr}(\cdot,\cdot)$: For the i-th query, only sample $k_i \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ when $i \ge i^*$ or $i \notin H$.
- $\mathsf{Eval}(\cdot,\cdot)$: For each query (S, f), do the same thing as Eval in $\mathsf{Hybrid}_4^{(N,Q)+1}$, except the generation of r_i. We generate r_i as follows. For each $i \in S$,
  - if $i < i^*$ and $i \in H$, let $r_i = \mathsf{PRF}_i.\mathsf{F}(S\|f)$.
  - If $i = i^*$ and $i^* \in H$, let $r_i \leftarrow \mathcal{O}(S\|f)$.
  - If $i > i^*$ or $i \notin H$, $r_i = \mathsf{PRF.Eval}(k_i, (S\|f))$.

When $\mathcal{O}'$ is $\mathsf{PRF.Eval}(k,\cdot)$ for a uniform random PRF key k, the adversary $\mathcal{A}'$ simulates the environment of $\mathsf{Hybrid}_5^{i^*}$ for $\mathcal{A}$. Hence,

$$\Pr\left[k \leftarrow \{0,1\}^\lambda : \mathcal{A}'^{\mathsf{PRF.Eval}(k,\cdot)}(1^\lambda) = 1\right] = \Pr_{\mathsf{Hybrid}_5^{i^*}}\left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\right].$$

When $\mathcal{O}'$ is a random function $\mathsf{F}(\cdot)$, the adversary $\mathcal{A}'$ simulates the environment of $\mathsf{Hybrid}_5^{i^*}$ for $\mathcal{A}$. Hence,

$$\Pr\left[\mathcal{A}'^{\mathsf{F}(\cdot)}(1^\lambda) = 1\right] = \Pr_{\mathsf{Hybrid}_5^{i^*+1}}\left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1\right].$$

From the security of PRF, we derive that $\mathsf{Hybrid}_5^{i^*}$ and $\mathsf{Hybrid}_5^{i^*+1}$ are indistinguishable. □

Lemma 5.7. Hybrid_5^{N+1} *is identical to* Hybrid_6^1. *Moreover, there exists a negligible function* $\nu(\lambda)$ *such that*

$$\left| \Pr_{\mathsf{Hybrid}_6^{q^*}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right] - \Pr_{\mathsf{Hybrid}_6^{q^*+1}} \left[\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda) = 1 \right] \right| \leq \nu(\lambda).$$

Proof. We build the following distinguisher $\mathcal{D}$ for the semi-malicous MPC security. The adversary $\mathcal{D}$ invokes the adversary $\mathcal{A}^{\mathsf{Regstr}(\cdot,\cdot),\mathsf{Eval}(\cdot,\cdot)}(1^\lambda)$, where the oracle $\mathsf{Regstr}(\cdot,\cdot)$ is the same as in Hybrid_5^{N+1}, and the oracle $\mathsf{Eval}(\cdot,\cdot)$ is implemented as follows.

Let the q-th query be (S, f), the oracle $\mathsf{Eval}(\cdot,\cdot)$ performs the same executions as in Hybrid_5^{N+1}, except the generation of $(\widetilde{\mathsf{msg}_i}, \widetilde{p_i})$ is replaced as follows.

- If $\mathsf{Map}(S, f)$ is defined before, then let $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i\in S\cap H} \leftarrow \mathsf{Map}(S, f)$. Othwerwise,
  - If $q < q^*$, let $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i\in S\cap H} \leftarrow \mathsf{One.Sim}(1^\lambda, S\cap H, \{x_i, r_i\}_{i\in S\setminus H}, f, f(\{x_i\}_{i\in S}))$.
  - If $q = q^*$, query the challenger with the number of parties $|S|$, the inputs $\{x_i\}_{i\in S}$, the honest party subset $H \cap S$, the randomness for dishonest parties $\{r_i\}_{i\in S\setminus H}$, and obtains the challenge $\{\mathsf{msg}_i, p_i\}_{i\in S\cap H}$. Let $\{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i\in S\cap H} = \{\mathsf{msg}_i, p_i\}_{i\in S\cap H}$, and define $\mathsf{Map}(S, f) = \{\mathsf{msg}_i, p_i\}_{i\in S\cap H}$.
  - If $q > q^*$, for each $i \in S \cap H$, set fresh randomness r_i. Let $\widetilde{\mathsf{msg}_i} = \mathsf{One.Round}_1\left(1^\lambda, x_i, f; r_i\right)$, $\widetilde{p_i} = \mathsf{One.Round}_2(x_i, r_i, \{\widetilde{\mathsf{msg}_j}\}_{j\in S}, f)$, and define $\mathsf{Map}(S, f) = \{\widetilde{\mathsf{msg}_i}, \widetilde{p_i}\}_{i\in S\cap H}$.

When $\{\mathsf{msg}_i, p_i\}_{i\in S\cap H}$ is obtained from real world execution, with dishonest parties' random coins $\{r_i\}_{i\in S\setminus H}$, the distinguisher $\mathcal{D}$ simulates the environment of $\mathsf{Hybrid}_6^{q^*}$ for $\mathcal{A}$. Hence,

$$\Pr\left[\begin{array}{l} \forall i \in S\cap H, r_i \leftarrow \{0,1\}^* \\ \forall i\in S, \mathsf{msg}_i = \mathsf{One.Round}_1(1^\lambda, x_i; r_i), \\ p_i = \mathsf{One.Round}_2(x_i, r_i, \{\mathsf{msg}_j\}_{j\in S}) \end{array} : \mathcal{D}(1^\lambda, \{\mathsf{msg}_i, p_i\}_{i\in S}) = 1\right] = \Pr\left[\mathcal{D}(1^\lambda, \mathsf{Hybrid}_6^{q^*}) = 1\right]$$

When $\{\mathsf{msg}_i, p_i\}_{i\in S\cap H}$ is obtained from the ideal simulation, then the distinguisher $\mathcal{D}$ simulates the environment of $\mathsf{Hybrid}_6^{q^*+1}$ for $\mathcal{A}$. Hence,

$$\Pr\Big[\{\mathsf{msg}_i, p_i\}_{i\in S\cap H} \leftarrow \mathsf{Sim}(1^\lambda, S\cap H, \{x_i, r_i\}_{i\in S\setminus H}, f, f(\{x_i\}_{i\in S})) : \\ \mathcal{D}(1^\lambda, \{\mathsf{msg}_i, p_i\}_{i\in S}) = 1\Big] = \Pr\left[\mathcal{D}(1^\lambda, \mathsf{Hybrid}_6^{q^*+1}) = 1\right].$$

Hence, from the semi-malicious security of the MPC protocol, we derive that $\mathsf{Hybrid}_6^{q^*}$ and $\mathsf{Hybrid}_6^{q^*+1}$ are indistinguishable. □

Lemma 5.8. *There exists a negligible function* $\nu(\lambda)$ *such that* $\mathsf{SD}(\mathsf{Hybrid}_6^{Q+1}, \mathsf{Ideal}) \leq \nu(\lambda)$.

Proof. Similar to Lemma 5.5, this Lemma follows from the statistical public key indistinguishability. □

Combining Lemma 5.2 to Lemma 5.8, we finish the proof. □

References

[1] Afshar, A., Mohassel, P., Pinkas, B., Riva, B.: Non-interactive secure computation based on cut-and-choose. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 387–404. Springer, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_22

[2] Ananth, P., Jain, A., Jin, Z., Malavolta, G.: Multi-key fully-homomorphic encryption in theplain model. TCC (2020)

[3] Ananth, P., Jain, A., Jin, Z., Malavolta, G.: Multikey fhe in the plain model. Cryptology ePrint Archive, Report 2020/180 (2020), https://eprint.iacr.org/2020/180

[4] Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: 45th FOCS. pp. 166–175. IEEE Computer Society Press, Rome, Italy (Oct 17–19, 2004). https://doi.org/10.1109/FOCS.2004.20

[5] Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 120–129. IEEE Computer Society Press, Palm Springs, CA, USA (Oct 22–25, 2011). https://doi.org/10.1109/FOCS.2011.40

[6] Badrinarayanan, S., Garg, S., Ishai, Y., Sahai, A., Wadia, A.: Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 275–303. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017). https://doi.org/10.1007/978-3-319-70700-6_10

[7] Badrinarayanan, S., Jain, A., Ostrovsky, R., Visconti, I.: Non-interactive secure computation from one-way functions. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 118–138. Springer, Heidelberg, Germany, Brisbane, Queensland, Australia (Dec 2–6, 2018). https://doi.org/10.1007/978-3-030-03332-3_5

[8] Bartusek, J., Garg, S., Masny, D., Mukherjee, P.: Reusable two-round mpc from ddh. Cryptology ePrint Archive, Report 2020/170 (2020), https://eprint.iacr.org/2020/170

[9] Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 500–532. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78375-8_17

[10] Benhamouda, F., Lin, H.: Multiparty reusable non-interactive secure computation. Cryptology ePrint Archive, Report 2020/221 (2020), https://eprint.iacr.org/2020/221

[11] Brakerski, Z., Döttling, N.: Two-message statistically sender-private ot from lwe. In: Theory of Cryptography Conference. pp. 370–390. Springer (2018)

[12] Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 597–608. ACM Press, Scottsdale, AZ, USA (Nov 3–7, 2014). https://doi.org/10.1145/2660267.2660374

[13] Cash, D., Hofheinz, D., Kiltz, E.: How to delegate a lattice basis. Cryptology ePrint Archive, Report 2009/351 (2009), https://eprint.iacr.org/2009/351

[14] Chase, M., Dodis, Y., Ishai, Y., Kraschewski, D., Liu, T., Ostrovsky, R., Vaikuntanathan, V.: Reusable non-interactive secure computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 462–488. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_15

[15] Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg, Germany, Zurich, Switzerland (Feb 9–11, 2010). https://doi.org/10.1007/978-3-642-11799-2_22

[16] Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer, Heidelberg, Germany, San Diego, CA, USA (Feb 24–26, 2014). https://doi.org/10.1007/978-3-642-54242-8_4

[17] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 467–476. ACM Press, Palo Alto, CA, USA (Jun 1–4, 2013). https://doi.org/10.1145/2488608.2488667

[18] Garg, S., Srinivasan, A.: Garbled protocols and two-round MPC from bilinear maps. In: Umans, C. (ed.) 58th FOCS. pp. 588–599. IEEE Computer Society Press, Berkeley, CA, USA (Oct 15–17, 2017). https://doi.org/10.1109/FOCS.2017.60

[19] Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 468–499. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78375-8_16

[20] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press, New York City, NY, USA (May 25–27, 1987). https://doi.org/10.1145/28395.28420

[21] Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: Yao, A.C.C. (ed.) ICS 2010. pp. 230–240. Tsinghua University Press, Tsinghua University, Beijing, China (Jan 5–7, 2010)

[22] Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 555–564. ACM Press, Palo Alto, CA, USA (Jun 1–4, 2013). https://doi.org/10.1145/2488608.2488678

[23] Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC. pp. 469–477. ACM Press, Portland, OR, USA (Jun 14–17, 2015). https://doi.org/10.1145/2746539.2746576

[24] Gordon, S.D., Liu, F.H., Shi, E.: Constant-round MPC with fairness and guarantee of output delivery. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 63–82. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015). https://doi.org/10.1007/978-3-662-48000-7_4

[25] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.)

ACM CCS 2006. pp. 89–98. ACM Press, Alexandria, Virginia, USA (Oct 30 – Nov 3, 2006). https://doi.org/10.1145/1180405.1180418, available as Cryptology ePrint Archive Report 2006/309

[26] Guruswami, V., Umans, C., Vadhan, S.: Unbalanced expanders and randomness extractors from parvaresh–vardy codes. J. ACM **56**(4) (Jul 2009). https://doi.org/10.1145/1538902.1538904, https://doi.org/10.1145/1538902.1538904

[27] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg, Germany, Tallinn, Estonia (May 15–19, 2011). https://doi.org/10.1007/978-3-642-20465-4_23

[28] Lombardi, A., Quach, W., Rothblum, R.D., Wichs, D., Wu, D.J.: New constructions of reusable designated-verifier NIZKs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 670–700. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_22

[29] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012). https://doi.org/10.1007/978-3-642-29011-4_41

[30] Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49896-5_26

[31] O'Neill, A., Peikert, C., Waters, B.: Bi-deniable public-key encryption. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 525–542. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011). https://doi.org/10.1007/978-3-642-22792-9_30

[32] Quach, W., Wee, H., Wichs, D.: Laconic function evaluation and applications. In: Thorup, M. (ed.) 59th FOCS. pp. 859–870. IEEE Computer Society Press, Paris, France (Oct 7–9, 2018). https://doi.org/10.1109/FOCS.2018.00086

[33] Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg, Germany, Aarhus, Denmark (May 22–26, 2005). https://doi.org/10.1007/11426639_27

[34] Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press, Toronto, Ontario, Canada (Oct 27–29, 1986). https://doi.org/10.1109/SFCS.1986.25