Simultaneous-Message and Succinct Secure Computation: Reusable and Multiparty Protocols

Siddharth Agarwal* Abhishek Jain† Akshayaram Srinivasan‡ David J. Wu§

Abstract

Recently, Boyle, Jain, Servan-Schreiber, and Srinivasan (EUROCRYPT 2025) introduced the notion of simultaneous-message and succinct (SMS) secure computation. In an SMS protocol, after an initial sampling of a common reference string (CRS), two parties—Alice (with a *large* input) and Bob (with a small input)—can simultaneously exchange encodings of their private inputs and obtain additive shares of the output of a function evaluated over their inputs. The key requirement is *succinctness*: namely, the sizes of the CRS and each input encoding grow only polylogarithmically in the size of Alice's input and the function output. Boyle et al., and independently Abram, Malavolta, and Roy (STOC 2025), constructed SMS for all bounded-depth Boolean circuits from the plain learning with errors (LWE) assumption.

In this work, we extend the study of SMS along two new dimensions:

- **Reusable SMS:** In this setting, the same input encodings can be reused to compute multiple functions.
- **Multiparty SMS:** In the multiparty setting, we consider computations over one large input and *multiple* small inputs. Succinctness in this case means the size of the CRS and input encodings can grow with the total length of the small inputs (but polylogarithmically with the length of the long input and the size of the function output).

Assuming polynomial hardness of LWE (with a sub-exponential modulus-to-noise ratio), we construct reusable two-party SMS for all bounded-depth Boolean circuits with polylogarithmic communication. By additionally assuming indistinguishability obfuscation, we present a generic compiler from reusable two-party SMS to reusable multiparty SMS.

Our construction of reusable two-party SMS from LWE relies on a new "dual-use" technique where we reuse an LWE secret key between a lattice-based algebraic homomorphic MAC and a lattice-based homomorphic encryption scheme. This dual-use technique allows us to bootstrap a reusable SMS protocol for quadratic functions into one that supports arbitrary (bounded-depth) Boolean circuits. Along the way, we also show how to adapt a previous lattice-based algebraic homomorphic MAC based on ring LWE to obtain one based on the *plain* LWE assumption.

1 Introduction

Consider the following scenario involving three parties, Alice, Bob, and Charlie: Alice holds a large private input $\mathbf{x}_0$, Bob holds a small private input $\mathbf{x}_1$, and Charlie wishes to learn the output $f(\mathbf{x}_0, \mathbf{x}_1)$ of some function f evaluated on the inputs of Alice and Bob. There is a simple and communication-efficient insecure protocol for this task: Bob sends his input to Alice, who computes the function output and sends it to Charlie. Is it possible to design a *secure* protocol that achieves, to the extent possible, the communication efficiency and interaction pattern of this insecure protocol?

Simultaneous-message and succinct secure computation. To address this question, a recent work of Boyle et al. [BJSS25] proposed the notion of *simultaneous-message and succinct* (SMS) secure computation. SMS is defined in the common reference string (CRS) model. To compute a function f, the parties proceed in two stages of simultaneous communication:

- **Encode:** Given f, Alice and Bob encode their private inputs and exchange their encodings with each other.

*University of Toronto. Email: siddharth@cs.toronto.edu.
†NTT Research and JHU. Email: abhishek@cs.jhu.edu.
‡University of Toronto. Email: akshayaram@cs.toronto.edu.
§UT Austin. Email: dwu4@cs.utexas.edu.

- **Decode:** Upon receiving the input encodings, Alice and Bob compute *additive* shares of the function output and send them to Charlie. Upon receiving the shares, Charlie sums the shares to obtain the function output.

The key requirement is *succinctness*: The size of the CRS and of the input encodings only grows polylogarithmically in the size of Alice's (large) input and the function output.[1] Furthermore, we require simulation-based security in the semi-honest model against collusion between either of the input parties (Alice or Bob) and the output party (Charlie). Compared to the insecure protocol, SMS necessarily requires two additional messages—one from Alice to Bob, and another from Bob to Charlie.[2] Furthermore, the total communication to Charlie is only twice the function output length. Thus, the communication overhead of SMS relative to the insecure protocol is essentially minimal.

SMS extends the notion of private simultaneous messages [FKN94] to require succinct communication and security against collusions. Furthermore, it subsumes the notion of trapdoor hash functions [DGI+19], which has applications to high-rate private information retrieval [CGKS95, KO97, DGI+19], correlation-intractable hash functions [CGH04, BKM20, JJ21], and hidden-bits generators [BCD+25]. It also bears similarity, but is incomparable to existing notions in succinct cryptography, including fully-homomorphic encryption (FHE) [Gen09], laconic function evaluation (LFE) [QWW18, CDG+17], and homomorphic secret sharing (HSS) [BGI16, DHRW16]. Finally, SMS can be used as a "rate-booster" for cryptographic primitives: for instance, it can be used to generically compile any FHE scheme into a rate-1 FHE scheme [BDGM19, BJSS25].

Assuming learning with errors (LWE) [Reg05], Boyle et al. [BJSS25] constructed an SMS scheme for all bounded-depth Boolean circuits where the input encoding size is polylogarithmic in the size of Alice's input (i.e., the long input) and slightly sublinear in the size of the function output. In an independent work, Abram et al. [AMR25] achieved polylogarithmic efficiency in both Alice's input and the output size under the same assumption. Assuming circular-secure LWE [HLL23], these results can be extended to circuits of unbounded depth.

This work: reusable and multiparty SMS. In this work, we extend the study of SMS along two new dimensions: (1) reusability of users' input encodings, and (2) computations over inputs involving more than two parties. In the following, we expand on each of these directions:

- **Reusable SMS:** In SMS, the input encodings of the parties depend on the function f they are computing. This means that if the parties wish to later compute a different function f' on the same inputs, they must recompute the encodings. This is in contrast to notions such as FHE and HSS that allow for homomorphic evaluations of different functions on the same ciphertext or shares. We address this shortcoming by proposing the notion of *reusable SMS*, where input encodings can be *reused* for computing multiple functions. More specifically, in reusable SMS, the parties execute the encoding stage given only their private input and the CRS. Later, given the description of any function f, they can compute shares of their output using the (same) input encodings.

- **Multiparty SMS:** SMS only supports computations over inputs of two parties. In this work, we extend SMS to the *multiparty* setting where one party holds a large input $\mathbf{x}_0$ and the other parties hold small inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Given a function f, the goal is to compute the function output $f(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$. The communication model and the efficiency requirements of multiparty SMS are the same as in two-party SMS: namely, the size of the input encodings should scale polylogarithmically with the length of the long input and the length of the function output, but may scale with the total length of the short inputs. We require semi-honest security against collusions between any subset of the input parties and the output party.

Reusable SMS vs. succinct HSS. Reusable SMS is closely related to the notion of succinct HSS [ARS24] and shares the key requirements of simultaneous communication, succinctness in the size of one of the inputs, additive output reconstruction, and importantly, reusability of input encodings. The key difference is that succinct HSS is defined in the correlated setup model where a trusted party samples a common public key which is used by the input holders to encode their inputs. In addition, the trusted party also samples *correlated* evaluation keys for the parties that are used to compute output shares. Reusable SMS, in contrast, only requires sampling a CRS without any additional correlated setup. In this sense, reusable SMS can be viewed as a strengthening of succinct HSS. Finally, we note that the best known constructions of succinct HSS with reusable input shares only support a limited class of computations of the form $\langle f(\mathbf{x}_0), g(\mathbf{x}_1)\rangle$,

[1] Due to information-theoretic lower bounds, we cannot expect to achieve succinctness in the input size of both parties.
[2] These extra messages are necessary to prevent input resetting attacks [HLP11].

where $\langle \cdot, \cdot \rangle$ denotes an inner product, f is an arbitrary polynomial-size circuit, and g is any logarithmic-depth circuit [CHP25, ILL25a]. In this work, we support arbitrary joint computations over all inputs. Reusable and multiparty SMS implies the notion of reusable two-round secure multiparty computation [BL20, AJJM20, BGMM20, AJJM21, BJKL21, BGSZ22] while additionally enjoying the same efficiency benefits as SMS relative to standard secure computation.

SMS vs. succinct cryptography. We also briefly compare the notion of SMS to other standard cryptographic notions for succinct multiparty computation:

- **Fully homomorphic encryption and laconic function evaluation:** Fully homomorphic encryption (FHE) [Gen09] allows one to securely compute a function where the communication complexity of the protocol is independent of the size of the circuit being computed. Moreover, the canonical two-message protocol based on FHE for computing over two private inputs[3] also achieves communication independent of the input size of one of the parties. However, compared to SMS, this protocol requires sequential (i.e., non-simultaneous) communication. For similar reasons, the dual notion of laconic function evaluation (LFE) [QWW18] also does not imply SMS (i.e., LFE supports succinct computation, but requires sequential communication).
- **Homomorphic secret sharing:** SMS is also incomparable to homomorphic secret sharing (HSS) [BGI16, DHRW16]. While SMS requires succinctness with respect to the size of one of the inputs, HSS only requires succinctness with respect to the size of the circuit. At the same time, HSS requires reusability of input shares, whereas plain SMS does not require reusability of input encodings for evaluation of different functions. One of our contributions in this work is augmenting plain SMS with reusability.
- **Trapdoor hash functions.** Finally, as observed in [BJSS25], SMS implies trapdoor hash functions [DGI+19].

1.1 Our Results

In this work, we define and construct the first reusable and multiparty SMS protocols. Throughout this work, we focus on SMS protocols for evaluating Boolean circuits.

Reusable two-party SMS. Our first result is a reusable two-party SMS protocol in the common *random* string model based on the LWE assumption.

Theorem 1.1 (Informal). *Assuming LWE (with a sub-exponential modulus-to-noise ratio), there exists a reusable two-party SMS for all bounded-depth Boolean circuits in the common random string model. Specifically, let λ be the security parameter. To evaluate a Boolean circuit $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}^m$ of depth d and size $\mathrm{poly}(\lambda) \le 2^\lambda$, our scheme achieves the following efficiency properties:*

- *The CRS has size $\mathrm{poly}(\lambda, d)$.*
- *The encoding of the large input $\mathbf{x}_0 \in \{0,1\}^N$ has size $\mathrm{poly}(\lambda, d)$ and the encoding of the short input $\mathbf{x}_1 \in \{0,1\}^k$ has size $\mathrm{poly}(\lambda, d, k)$.*[4]

Reusable multiparty SMS. By additionally assuming the existence of indistinguishability obfuscation ($i\mathcal{O}$) [BGI+01, GGH+13], we can extend the above result to the *multiparty* setting in the common *reference* string model. In fact, we present a general compiler from two-party reusable SMS to a multiparty reusable SMS, assuming $i\mathcal{O}$ together with a non-reusable (two-party) SMS scheme (where the decoding function of the party with the short input has a small circuit description). By instantiating both of the underlying SMS protocols from LWE, we obtain the following theorem:

[3] Concretely, in this protocol, Bob would first encrypt his input using FHE and send the encrypted input to Alice. Alice then homomorphically evaluates the function on Bob's encrypted input and her private input and sends back the encrypted result. Bob can decrypt and learn the output. Optionally, if we want the parties to obtain additive shares of the joint output, Alice can homomorphically apply a random shift to the output at the end of the homomorphic evaluation process.

[4] Since we always bound $|C| \le 2^\lambda$, this means $N, m \le 2^\lambda$. Thus we can absorb polylogarithmic dependencies in the input length N and the output length m into a $\mathrm{poly}(\lambda)$ term.

Theorem 1.2 (Informal)**.** *Assuming sub-exponentially secure iO for circuits and sub-exponentially secure LWE (with a sub-exponential modulus-to-noise ratio), there exists a reusable multiparty SMS scheme for bounded-depth Boolean circuits. Specifically, let λ be the security parameter. To evaluate a Boolean circuit $C\colon \{0,1\}^N \times (\{0,1\}^k)^n \to \{0,1\}^m$ of depth d and size $\mathsf{poly}(\lambda) \le 2^\lambda$, our scheme achieves the following efficiency properties:*

- *The CRS has size $\mathsf{poly}(\lambda, d, n, k)$.*
- *The encoding of the large input $\mathbf{x}_0 \in \{0,1\}^N$ has size $\mathsf{poly}(\lambda, d)$ and the encoding of the short inputs $\mathbf{x}_i \in \{0,1\}^k$ has size $\mathsf{poly}(\lambda, k)$.*
- *The evaluation time of parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ (with the short inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \{0,1\}^k$) is $\mathsf{poly}(\lambda, d, n, k)$,[5] and in particular, does not depend on the description length of C.*

2 Technical Overview

We start with the formal syntax of a *reusable* simultaneous-message and succinct (SMS) secure computation protocol. Specifically, we consider the two-party setting considered in [BJSS25] where one party (denoted $\mathcal{P}_0$) holds a long input $\mathbf{x}_0 \in \{0,1\}^N$ and the other party (denoted $\mathcal{P}_1$) holds a short input $\mathbf{x}_1 \in \{0,1\}^k$. A reusable SMS protocol then consists of three main algorithms:

- **Setup:** The setup algorithm Setup takes as input the security parameter λ and outputs a common reference string (CRS).
- **Input encoding:** Using the CRS, each party can use the Encode algorithm to encode their input (either $\mathbf{x}_0$ or $\mathbf{x}_1$) to obtain an input encoding pe_i and a decoding trapdoor td_i.
- **Circuit evaluation:** Finally, there is a local evaluation algorithm Eval that takes as input the CRS, the parties' input encodings $\mathsf{pe}_0, \mathsf{pe}_1$, the description of a Boolean circuit $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}^m$, along with a decoding trapdoor td_i (for $i \in \{0,1\}$). The local evaluation algorithm outputs a share d_i.

Moreover, the scheme should satisfy the following properties:

- **Correctness:** The correctness requirement says that if pe_0 is an encoding of $\mathcal{P}_0$'s input $\mathbf{x}_0 \in \{0,1\}^N$ and pe_1 is an encoding of $\mathcal{P}_1$'s input $\mathbf{x}_1 \in \{0,1\}^k$, then the associated shares d_0, d_1 output by Eval are an additive secret sharing of the output $C(\mathbf{x}_0, \mathbf{x}_1)$. Namely, we have that $d_0 \oplus d_1 = C(\mathbf{x}_0, \mathbf{x}_1)$.
- **Security:** The security requirement asserts that a party's public encoding pe_i should computationally hide their associated input (i.e., pe_0 computationally hides $\mathbf{x}_0$ and pe_1 computationally hides $\mathbf{x}_1$).
- **Succinctness:** The succinctness requirement is that the length of the encoding pe_0 of the long input $\mathbf{x}_0 \in \{0,1\}^N$ scales polylogarithmically with N and the output length m of the circuit C. The length of the encoding pe_1 of the short input $\mathbf{x}_1 \in \{0,1\}^k$ is allowed to scale polynomially with k, but also polylogarithmically with the output length m. For simplicity, throughout this work, we assume the size of the circuit C is at most 2^λ, so we can bound $\log N$ and $\log m$ by the security parameter λ. In this case, our efficiency requirements are

  $$|\mathsf{pe}_0| = \mathsf{poly}(\lambda) \quad \text{and} \quad |\mathsf{pe}_1| = \mathsf{poly}(\lambda, k).$$

  When considering bounded-depth computations (i.e., settings where the circuit C has depth at most d), we also allow the size of the encodings $\mathsf{pe}_0, \mathsf{pe}_1$ to scale with the depth bound d.

An important feature of the above definition is that the input encoding algorithm is *circuit-independent*. The circuit is only specified at evaluation time rather than encoding time. This means the same input encodings can be reused to evaluate many different circuits. This is similar to primitives like fully homomorphic encryption [Gen09] or homomorphic secret sharing [BGI16] where the same ciphertext or shares can be reused across multiple computations. Thus, we say the SMS scheme is reusable. In the original notion of SMS from [BJSS25], the encoding function depended also on the circuit being computed. To evaluate different circuits over the same input, the parties would have to re-encode their input each time. In this work, we focus on the more versatile notion of *reusable* SMS.

[5] As in Theorem 1.1, we absorb all polylogarithmic dependencies on the circuit size $|C|$ into the $\mathsf{poly}(\lambda)$ term.

2.1 Reusable Two-Party SMS from LWE

Our first contribution in this work is a construction of a *reusable* two-party SMS scheme from LWE. Our construction relies on a new "dual-use" technique of composing an algebraic homomorphic MAC based on lattices introduced in the recent work of Ishai, Li, and Lin [ILL25a, ILL25b] together with a lattice-based homomorphic encryption scheme. Namely, we consider a setting where the LWE secret key is *reused* across both an algebraic homomorphic MAC and a (leveled) homomorphic encryption scheme. In a very different setting, the work of [BTVW17] described a dual-use technique of sharing an LWE secret key across a lattice-based attribute-based encryption scheme and a homomorphic encryption scheme. They showed how this composition enables new approaches for predicate encryption and private constrained PRFs.

Algebraic homomorphic MACs. In an algebraic homomorphic MAC [ILL25a, ILL25b] over $\mathbb{Z}_p^n$, a tag $\boldsymbol{\sigma} \in \mathbb{Z}_p^n$ on an input bit $x \in \{0,1\}$ satisfies $\boldsymbol{\sigma} = x \cdot \mathbf{s} + \mathbf{k} \in \mathbb{Z}_p^n$, where we refer to $\mathbf{s} \in \mathbb{Z}_p^n$ as a "global offset" and $\mathbf{k} \in \mathbb{Z}_p^n$ is a (pseudorandom) blinding term (which we often refer to as the "encoding key" associated with $\boldsymbol{\sigma}$). An algebraic homomorphic MAC allows one to homomorphically operate over tags encoded with respect to the *same* global offset $\mathbf{s}$ (and arbitrary encoding keys). We consider a setting where there are two global offsets $\mathbf{s}_{\mathsf{in}}, \mathbf{s} \in \mathbb{Z}_p^n$, where $\mathbf{s}_{\mathsf{in}}$ is the global offset associated with input values and $\mathbf{s}$ is the global offset associated with the output of a homomorphic computation. Then, an algebraic homomorphic MAC consists of three main algorithms:

- **Setup:** The setup algorithm Setup takes the output global offset $\mathbf{s} \in \mathbb{Z}_p^n$ as input and outputs a public evaluation key evk, a secret key sk, and a global offset $\mathbf{s}_{\mathsf{in}} \in \mathbb{Z}_p^n$ for input values. Note that we provide the output offset $\mathbf{s}$ as an *explicit* input because we will consider scenarios where the *same* $\mathbf{s}$ is reused across the algebraic homomorphic MAC and a leveled homomorphic encryption scheme. This "dual-use" of the offset $\mathbf{s}$ is a central technique introduced in this work.
- **Tag computation:** Take any Boolean circuit $C\colon \{0,1\}^\ell \to \{0,1\}^m$ and input $\mathbf{x} \in \{0,1\}^\ell$. Suppose we have a collection of tags $\boldsymbol{\sigma}_i = x_i \cdot \mathbf{s}_{\mathsf{in}} + \mathbf{k}_i \in \mathbb{Z}_p^n$ encoded with respect to the input offset $\mathbf{s}_{\mathsf{in}}$ and *arbitrary* encoding keys $\mathbf{k}_1, \ldots, \mathbf{k}_\ell$. We will often write this more compactly as

  $$\boldsymbol{\sigma}_{\mathbf{x}} = (\mathbf{x} \otimes \mathbf{s}_{\mathsf{in}}) + \mathbf{k} \in \mathbb{Z}_p^{n\ell}.$$

  Then, the *public* evaluation algorithm $\mathsf{EvalTag}(\mathsf{evk}, C, \mathbf{x}, \boldsymbol{\sigma}_{\mathbf{x}})$ outputs a tag $\boldsymbol{\sigma}_{\mathbf{y}} \in \mathbb{Z}_p^{nm}$ on the output $\mathbf{y} = C(\mathbf{x})$ with respect to the global offset $\mathbf{s}$. Namely,

  $$\boldsymbol{\sigma}_{\mathbf{y}} = (C(\mathbf{x}) \otimes \mathbf{s}) + \mathbf{k}_C \in \mathbb{Z}_p^{nm}. \tag{2.1}$$

  A critical property of algebraic homomorphic MACs is the output encoding key $\mathbf{k}_C \in \mathbb{Z}_p^{nm}$ only depends on the circuit C and does *not* depend on the input $\mathbf{x}$. In fact, there is a separate key-computation algorithm (see below) that computes $\mathbf{k}_C$ from the input encoding key $\mathbf{k}$ and the circuit C.
- **Key computation:** Using the secret key sk, the key-computation algorithm $\mathsf{EvalKey}(\mathsf{sk}, C, \mathbf{k})$ computes the output encoding key $\mathbf{k}_C$ satisfying Eq. (2.1). Importantly, this only requires knowledge of the key $\mathbf{k}$ associated with the input and the description of the circuit C. Moreover, Eq. (2.1) holds for *every* input encoding key $\mathbf{k}$ and tag $\boldsymbol{\sigma}_{\mathbf{x}} = (\mathbf{x} \otimes \mathbf{s}_{\mathsf{in}}) + \mathbf{k}$.

We can also view the pair $(\boldsymbol{\sigma}_{\mathbf{y}}, \mathbf{k}_C)$ in Eq. (2.1) as a *secret sharing* of the value $\mathbf{y} \otimes \mathbf{s} = C(\mathbf{x}) \otimes \mathbf{s}$ over $\mathbb{Z}_p^{nm}$. Thus, an algebraic homomorphic MAC allows two parties (one holding tags $\boldsymbol{\sigma}_{\mathbf{x}} \in \mathbb{Z}_p^{n\ell}$ on an input $\mathbf{x}$ and the other holding the associated encoding key $\mathbf{k} \in \mathbb{Z}_p^{n\ell}$ and the secret key sk) to *non-interactively* compute a secret sharing of $C(\mathbf{x}) \otimes \mathbf{s}$. This observation lends itself naturally to the structure of an SMS protocol.

A template for building SMS from an algebraic homomorphic MAC. We now describe a basic template for constructing a two-party SMS protocol using an algebraic MAC. This first template is completely insecure, but illustrates the conceptual ideas underlying our design. Throughout this section, we will assume Alice holds the long input $\mathbf{x}_0 \in \{0,1\}^N$ and Bob holds the short input $\mathbf{x}_1 \in \{0,1\}^k$. Since the input encodings are *reusable*, we can without loss of generality consider evaluating a circuit with a single output bit. We can handle circuits with m output bits by evaluating circuits $C_1, \ldots, C_m$, where C_i computes the i^{th} output bit.

- **Setup:** First, suppose that Alice has an algebraic homomorphic MAC on her input $\boldsymbol{\sigma}_{\mathbf{x}_0} = (\mathbf{x}_0 \otimes \mathbf{s}_{\mathsf{in}}) + \mathbf{k}_0 \in \mathbb{Z}_p^{Nn}$ and Bob has an algebraic homomorphic MAC on his input $\boldsymbol{\sigma}_{\mathbf{x}_1} = (\mathbf{x}_1 \otimes \mathbf{s}_{\mathsf{in}}) + \mathbf{k}_1 \in \mathbb{Z}_p^{kn}$. Suppose also that Bob knows the secret key sk for the algebraic homomorphic MAC as well as the encoding keys $\mathbf{k}_0$ and $\mathbf{k}_1$.

- **Input encoding:** Bob's public input encoding will simply be the algebraic homomorphic MAC on his input $\boldsymbol{\sigma}_{\mathbf{x}_1}$ together with his input $\mathbf{x}_1$. (This is of course insecure, but can be fixed below by replacing $\mathbf{x}_1$ with an encryption of $\mathbf{x}_1$). For now, Alice does not provide any input encoding. By construction, this approach satisfies our efficiency requirements.

- **Circuit evaluation:** Given a circuit $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}$, Alice now applies EvalTag on the circuit C, the inputs $(\mathbf{x}_0, \mathbf{x}_1)$, and the tags $(\boldsymbol{\sigma}_{\mathbf{x}_0}, \boldsymbol{\sigma}_{\mathbf{x}_1})$ to obtain a tag $\boldsymbol{\sigma}_{\mathbf{y}}$ on the output $\mathbf{y} = C(\mathbf{x}_0, \mathbf{x}_1)$. Similarly, Bob applies EvalKey on the circuit C and the encoding keys $(\mathbf{k}_0, \mathbf{k}_1)$ to obtain a key $\mathbf{k}_C$. The correctness guarantee of the algebraic homomorphic MAC now ensures that

  $$\boldsymbol{\sigma}_{\mathbf{y}} = (C(\mathbf{x}_0, \mathbf{x}_1) \otimes \mathbf{s}) + \mathbf{k}_C = C(\mathbf{x}_0, \mathbf{x}_1) \cdot \mathbf{s} + \mathbf{k}_C \in \mathbb{Z}_p^n.$$

  In particular, Alice and Bob now have a secret sharing of $C(\mathbf{x}_0, \mathbf{x}_1) \cdot \mathbf{s}$. Suppose that we set the first component of the global offset $\mathbf{s}$ to be 1 (i.e., $s_1 = 1$). Then, the first component of $\boldsymbol{\sigma}_{\mathbf{y}}$ together with the first component of $\mathbf{k}_C$ represent a secret sharing of $C(\mathbf{x}_0, \mathbf{x}_1)$.

This basic template satisfies our correctness and efficiency guarantees. Notably, the algebraic homomorphic MAC ensures that the two parties' output is an additive secret sharing of the output. However, there are two major problems with this template:

- **Generating a homomorphic MAC on Alice's input.** First, we are assuming that Alice has an algebraic homomorphic MAC $\boldsymbol{\sigma}_{\mathbf{x}_0}$ on her input. It is not clear how she computes this (without knowledge of the secret key). On the other hand, Bob knows the secret key for the algebraic homomorphic MAC, so he can compute his tag $\boldsymbol{\sigma}_{\mathbf{x}_1}$ himself.

- **Hiding Bob's input.** The second major problem is the algebraic homomorphic MAC only supports computations on *public* inputs. Namely, the EvalTag operation requires knowledge of the input $\mathbf{x}$ associated with the tag $\boldsymbol{\sigma}_{\mathbf{x}}$. In the above template, this means Bob had to reveal his input to Alice in order for Alice to compute on it. Thus, to satisfy the security requirement of an algebraic MAC, we need a way to hide Bob's input.

Below, we show how to address each of these challenges. We handle the first challenge by using a reusable SMS protocol for quadratic polynomials (which was recently constructed in the work of [AMR25]), and tackle the second challenge by having Bob encrypt his input and then employing a new *dual-use* technique that exploits the algebraic structures of lattice-based homomorphic encryption and algebraic homomorphic MACs.

From public to *private* computation. To hide Bob's input, the natural approach is to *encrypt* it using a (homomorphic) encryption scheme. Consider the following approach:

- We take the basic SMS template from above, but now, instead of Bob sending his input $\mathbf{x}_1 \in \{0,1\}^k$ in the clear, Bob will first encrypt it using a (homomorphic) encryption scheme. Let $\mathbf{c}_1 \in \{0,1\}^t$ be the encryption of $\mathbf{x}_1$. In the SMS protocol, Bob now publishes the ciphertext $\mathbf{c}_1$ together with the tag $\boldsymbol{\sigma}_{\mathbf{c}_1}$ as part of his public input encoding.

- As before, let $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}^m$ be the circuit the two parties want to compute. Let f_C be the Boolean circuit that takes as input $\mathbf{x}_0 \in \{0,1\}^N$ and a ciphertext $\mathbf{c} \in \{0,1\}^t$, and homomorphically evaluates the circuit $C(\mathbf{x}_0, \cdot)$ on the ciphertext $\mathbf{c}$. Namely, $f_C(\mathbf{x}_0, \mathbf{c}_1)$ outputs the binary representation of a ciphertext that decrypts to $C(\mathbf{x}_0, \mathbf{x}_1)$ under the public key pk. Let $\mathbf{d} = f_C(\mathbf{x}_0, \mathbf{c}_1)$.

- Suppose Alice now evaluates the Boolean circuit computing the binary decomposition of f_C on the tags $(\boldsymbol{\sigma}_{\mathbf{x}_0}, \boldsymbol{\sigma}_{\mathbf{c}_1})$. Let $\boldsymbol{\sigma}_{\mathbf{d}}$ be the resulting tag. The correctness property of the algebraic homomorphic MAC now says that

  $$\boldsymbol{\sigma}_{\mathbf{d}} = (\mathbf{d} \otimes \mathbf{s}) + \mathbf{k}_{f_C}. \tag{2.2}$$

Observe that this computation only requires knowledge of the ciphertext $\mathbf{c}$ (and the public key for the homomorphic encryption scheme), and does not *require* knowledge of Bob's input $\mathbf{x}_1$. Similarly, Bob can compute the new encoding key $\mathbf{k}_{f_C}$ given knowledge of the input encoding keys $\mathbf{k}_0, \mathbf{k}_1$ and the description of the circuit computing f_C (which does *not* require knowledge of Alice's input $\mathbf{x}_0$).

While this template addresses the security issue of Bob having to reveal his input $\mathbf{x}_1$, it introduces a new problem. At the end of the protocol, Alice and Bob have a secret sharing of $\mathbf{d} \otimes \mathbf{s}$, where $\mathbf{s}$ is the global offset associated with the algebraic homomorphic MAC, and $\mathbf{d}$ is the binary representation of an encryption of $C(\mathbf{x}_0, \mathbf{x}_1)$, *not* the actual value of $C(\mathbf{x}_0, \mathbf{x}_1)$, which is what we are actually trying to recover.

The dual-use technique. In order to recover $C(\mathbf{x}_0, \mathbf{x}_1)$ from $\mathbf{d}$, we need to implement *decryption*. The crux of the dual-use technique is that the decryption operation in many lattice-based homomorphic encryption schemes (e.g., [BV11, BGV12, Bra12, GSW13]) is nearly linear. Namely, a ciphertext (encrypting a bit $\mu \in \{0,1\}$) can be taken to be a vector $\mathbf{c} \in \mathbb{Z}_p^n$, the secret key is also a vector $\mathbf{s} \in \mathbb{Z}_p^n$, and the decryption algorithm simply computes $\mathbf{s}^\intercal \mathbf{c} \bmod p$. This yields a noisy "encoding" of the message:

$$\mathbf{s}^\intercal \mathbf{c} = \mu \cdot \lfloor p/2 \rfloor + e \in \mathbb{Z}_p,$$

where e is a small error term. To recover μ, one scales the result by $2/p$ and then rounds to the nearest integer.

Suppose now that the global offset $\mathbf{s} \in \mathbb{Z}_p^n$ in the algebraic homomorphic MAC was *also* the decryption key for the homomorphic encryption scheme, and the homomorphic encryption scheme had a nearly linear decryption procedure. In this case, we can recover shares of $C(\mathbf{x}_0, \mathbf{x}_1)$ from shares of $\mathbf{d} \otimes \mathbf{s}$. To see this, suppose $\tilde{\mathbf{d}} \in \mathbb{Z}_p^n$ is an encryption of $C(\mathbf{x}_0, \mathbf{x}_1)$ and let $\mathbf{d}$ be its binary decomposition. This means

$$\mathbf{s}^\intercal \tilde{\mathbf{d}} = C(\mathbf{x}_0, \mathbf{x}_1) \cdot \lfloor p/2 \rfloor + e \in \mathbb{Z}_p,$$

where e is small error. For ease of notation, let $\ell = \lceil \log p \rceil$. Then, we can write $\mathbf{d}^\intercal = [d_{1,1}, \ldots, d_{1,\ell}, \ldots, d_{n,1}, \ldots, d_{n,\ell}]$ where $(d_{i,1}, \ldots, d_{i,\ell})$ is the binary representation of $\tilde{d}_i \in \mathbb{Z}_p$. In this case, given $\mathbf{d} \otimes \mathbf{s} \in \mathbb{Z}_p^{n^2\ell}$, we can apply a linear map $L \colon \mathbb{Z}_p^{n^2\ell} \to \mathbb{Z}_p$ such that

$$L(\mathbf{d} \otimes \mathbf{s}) = \sum_{i \in [n]} \sum_{j \in [\ell]} 2^{\ell - j} s_i d_{i,j} = \sum_{i \in [n]} s_i \tilde{d}_i = \mathbf{s}^\intercal \tilde{\mathbf{d}} = C(\mathbf{x}_0, \mathbf{x}_1) \cdot \lfloor p/2 \rfloor + e \in \mathbb{Z}_p, \tag{2.3}$$

where e is a small error term. Since L is a linear function, we can combine it with the algebraic homomorphic MAC invariant from Eq. (2.2) and conclude that

$$L(\boldsymbol{\sigma}_{\mathbf{d}}) = L(\mathbf{d} \otimes \mathbf{s} + \mathbf{k}_{f_C}) = C(\mathbf{x}_0, \mathbf{x}_1) \cdot \lfloor p/2 \rfloor + e + L(\mathbf{k}_{f_C}).$$

In particular, the pair $(L(\boldsymbol{\sigma}_{\mathbf{d}}), L(\mathbf{k}_{f_C}))$ is an additive share of $C(\mathbf{x}_0, \mathbf{x}_1) \cdot \lfloor p/2 \rfloor + e$. At this point, we can use the standard local rounding techniques from [DHRW16, BKS19] to have Alice and Bob locally round their respective shares and recover additive shares of $C(\mathbf{x}_0, \mathbf{x}_1)$ over $\mathbb{Z}_2$.[6]

To summarize, by having the homomorphic encryption scheme and the algebraic homomorphic MAC share a common $\mathbf{s}$ (as the decryption key and the output global offset), the algebraic homomorphic MAC evaluation process is implicitly performing a homomorphic evaluation followed by a homomorphic decryption procedure. This allows Alice and Bob to transform secret shares on their input to secret shares of the *decrypted* output.

Bootstrapping from an SMS scheme for degree-two functions. The remaining wrinkle in our basic template above is our protocol assumes that Alice already has an algebraic MAC on her input $\mathbf{x}_0$ while Bob holds the secret key for the algebraic homomorphic MAC. Thus, we need a mechanism for Alice to learn the tag $\boldsymbol{\sigma}_{\mathbf{x}_0}$ and Bob to learn the associated encoding key $\mathbf{k}_0$ where

$$\boldsymbol{\sigma}_{\mathbf{x}_0} = (\mathbf{x}_0 \otimes \mathbf{s}_{\mathsf{in}}) + \mathbf{k}_0 \in \mathbb{Z}_p^{Nn}. \tag{2.4}$$

[6] Specifically, the parties apply a random shift $u \xleftarrow{R} \mathbb{Z}_p$ to the output (where u is sampled as part of the CRS), scale the result by $2/p$, and then round to the nearest integer. For appropriate choices of p, q, the works of [DHRW16, BKS19] show that with overwhelming probability over the choice of u, the result will be an additive share of $C(\mathbf{x}_0, \mathbf{x}_1)$ over $\mathbb{Z}_2$. We refer to the proof of Theorem 5.2 for the full analysis.

Equivalently, we can view the pair $(\boldsymbol{\sigma}_{\mathbf{x}_0}, \mathbf{k}_0)$ as an additive secret sharing of $\mathbf{x}_0 \otimes \mathbf{s}_{\mathsf{in}}$. Thus, an equivalent way of formulating the problem is we need an efficient mechanism for Alice and Bob to derive secret shares of the value $\mathbf{x}_0 \otimes \mathbf{s}_{\mathsf{in}}$. In this case, Alice holds the long input $\mathbf{x}_0$ while Bob holds the short input $\mathbf{s}_{\mathsf{in}}$. Observe now that this is essentially an SMS problem for a *degree-2* computation and where we want the outputs to be an additive secret sharing over $\mathbb{Z}_p^{Nn}$ (as opposed to a secret sharing over $\mathbb{Z}_2$). The recent work of [AMR25] constructs a notion called a "reverse trapdoor hash" from the learning with errors assumption, which can be adapted to obtain a reusable two-party SMS for degree-2 functions where the output shares $(\boldsymbol{\sigma}_{\mathbf{x}_0}, \mathbf{k}_0)$ precisely satisfy Eq. (2.4). Using this scheme, we now have an efficient mechanism for Alice to derive the tag $\boldsymbol{\sigma}_{\mathbf{x}_0}$ on her input.

To summarize, our final reusable SMS protocol can be viewed as a way to bootstrap a reusable SMS protocol for degree-2 polynomials into one that supports arbitrary bounded-depth Boolean circuits. The bootstrapping step relies on a dual-use combination of lattice-based algebraic homomorphic MACs with lattice-based (leveled) homomorphic encryption. Putting all the pieces together, our final protocol operates as follows:

- **Setup:** The CRS in our scheme consists of the CRS for the underlying SMS for quadratic functions as well as the local rounding randomness (used for local decryption).
- **Alice's input encoding:** Alice's input encoding consists of an encoding for the underlying SMS scheme on her (long) input $\mathbf{x}_0$. Note that the resulting input encoding is short by succinctness of the underlying SMS for quadratic functions.
- **Bob's input encoding:** Bob samples a secret key $\mathbf{s}$ for the homomorphic encryption scheme and encrypts his input $\mathbf{x}_1$ with the secret key $\mathbf{s}$. Let $\mathbf{c}_1$ be the resulting ciphertext. Next, Bob samples the public evaluation key evk for the algebraic homomorphic MAC with $\mathbf{s}$ as the global offset (associated with the outputs). Let $\mathbf{s}_{\mathsf{in}}$ be the offset associated with the inputs. Bob's input encoding now consists of the evaluation key evk, the encryption $\mathbf{c}_1$ of his input, the tag $\boldsymbol{\sigma}_{\mathbf{c}_1}$ on the ciphertext $\mathbf{c}_1$, as well as the SMS encoding on the input $\mathbf{s}_{\mathsf{in}}$.
- **Circuit evaluation:** To evaluate a circuit $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}$ on the input encodings, Alice and Bob perform the following sequence of *local* operations:
  1. Alice and Bob first *locally* run the SMS protocol for quadratic functions on the function $g(\mathbf{x}_0, \mathbf{s}_{\mathsf{in}}) \coloneqq \mathbf{x}_0 \otimes \mathbf{s}_{\mathsf{in}}$ Let $\boldsymbol{\sigma}_{\mathbf{x}_0}$ be Alice's share and $\mathbf{k}_0$ be Bob's share. Correctness of the underlying SMS protocol says that the local shares $\boldsymbol{\sigma}_{\mathbf{x}_0}$ and $\mathbf{k}_0$ precisely satisfy Eq. (2.4). Hence, we can view $\boldsymbol{\sigma}_{\mathbf{x}_0}$ as an algebraic homomorphic MAC on $\mathbf{x}_0$ with respect to the encoding key $\mathbf{k}_0$ (and input offset $\mathbf{s}_{\mathsf{in}}$).
  2. At this point, Alice can homomorphically evaluate the circuit f_C on the tags $(\boldsymbol{\sigma}_{\mathbf{x}_0}, \boldsymbol{\sigma}_{\mathbf{c}_1})$, where f_C is the circuit that takes input $(\mathbf{x}_0, \mathbf{c}_1)$ and homomorphically evaluates the circuit $C(\mathbf{x}_0, \cdot)$ on the ciphertext $\mathbf{c}_1$. Similarly, Bob homomorphically evaluates f_C on the encoding keys $(\mathbf{k}_0, \mathbf{k}_1)$.
  3. Next, Alice and Bob locally apply the linear function L from Eq. (2.3) to their respective shares. By the dual-use technique, Alice and Bob at this point have derived an additive secret sharing of $C(\mathbf{x}_0, \mathbf{x}_1) \cdot \lfloor p/2 \rfloor + e \in \mathbb{Z}_p$.
  4. Finally, using the local rounding procedure [DHRW16, BKS19], Alice and Bob recover additive shares of $C(\mathbf{x}_0, \mathbf{x}_1)$ over $\mathbb{Z}_2$, as desired.

Security of our protocol directly reduces to security of the underlying schemes. In particular, Alice's input encoding hides her input by security of the SMS scheme for quadratic functions. Security for Bob's encoding follows from SMS security of the underlying SMS scheme (to argue that $\mathbf{s}_{\mathsf{in}}$ is hidden), security of the algebraic homomorphic MAC (to argue that the output mask $\mathbf{s}$ is hidden), and security of the homomorphic encryption scheme (to argue that the ciphertext $\mathbf{c}_1$ hides $\mathbf{x}_1$). We provide the full details of our construction in Section 5.

Instantiating the primitives from LWE. Finally, we note that all of the building blocks from the above construction can be instantiated from the plain LWE assumption. As noted previously, the work of [AMR25] can be leveraged to obtain a two-party reusable SMS for quadratic functions from LWE and the homomorphic encryption scheme with nearly linear decryption can be instantiated from most lattice-based homomorphic encryption schemes [BV11, BGV12, Bra12, GSW13]. For the algebraic homomorphic MAC, the recent work of [ILL25a, ILL25b] describes a construction based on ring LWE (that supports all bounded-depth Boolean circuits). It is straightforward to adapt their construction to one based on plain LWE. For completeness, we provide a formal description of this adaptation in this work (see Section 7).

2.2 A Generic Approach to Multiparty SMS via Indistinguishability Obfuscation

The construction described in Section 2.1 gives a reusable two-party SMS protocol for arbitrary bounded-depth Boolean circuit families from the plain LWE assumption. A natural question is whether we can generalize this to a *multiparty* SMS protocol. In this setting, there are $n+1$ parties $\mathcal{P}_0, \ldots, \mathcal{P}_n$ where party $\mathcal{P}_0$ holds a long input $\mathbf{x}_0 \in \{0,1\}^N$ and the remaining parties $\mathcal{P}_i$ for $i \in [n]$ hold short inputs $\mathbf{x}_i \in \{0,1\}^k$. As in the two-party case, each party can publish an encoding of their respective input. Later, given an arbitrary Boolean circuit $C\colon \{0,1\}^N \times (\{0,1\}^k)^n \to \{0,1\}$, each party can locally compute an additive share of the output $C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$. We require that the size of the CRS and the size of the input encodings be polylogarithmic in the size of the long input $\mathbf{x}_0$ as well as the output length of the family of circuits.

Constructing multiparty SMS is much more challenging. For instance, several of the ideas underlying the two-party SMS protocol from Section 2.1 do not appear to extend to more than two parties. These include the dual-use technique of composing the algebraic homomorphic MAC with leveled homomorphic encryption as well as the local rounding procedure needed to transform partial decryptions into shares of the underlying message. In the non-input-succinct regime, the work of [DHRW16] showed how to lift a two-party spooky encryption scheme to a multiparty protocol with additive reconstruction via a GMW [GMW87] style gate-by-gate computation procedure. However, this does not work in our input-succinct setting. While the lattice techniques do not directly extend to the multiparty setting, we show that if we rely on indistinguishability obfuscation, then we can *generically* lift any two-party SMS protocol into a multiparty SMS protocol.

From two-party SMS to multiparty SMS. The main idea underlying our construction is to delegate the computation of $\mathcal{P}_0$'s share to an obfuscated program in the CRS. The main challenge is to do so in a way that preserves succinctness of the CRS. This latter step will rely on the underlying two-party reusable SMS protocol. To illustrate this idea, we begin with a straw-man scheme that has a *long* CRS.

- **Setup:** First, the CRS will contain a public key pk_i for each party $\mathcal{P}_i$ for each $i \in [n]$. In addition, it will also contain an obfuscated program that has the associated decryption keys sk_i hardcoded inside. Essentially, the obfuscated program takes as input the (short) encodings from all the parties and outputs the share for $\mathcal{P}_0$. We provide more details on the precise behavior below.
- **Party $\mathcal{P}_0$'s encoding:** In the straw-man scheme, party $\mathcal{P}_0$'s input encoding pe_0 is a succinct commitment to their input $\mathbf{x}_0 \in \{0,1\}^N$.
- **Party $\mathcal{P}_i$'s encoding:** Party $\mathcal{P}_i$ (for $i \in [n]$) encodes their input $\mathbf{x}_i \in \{0,1\}^k$ by first sampling a key k_i for a pseudorandom function (PRF) and then encrypting the pair $(k_i, \mathbf{x}_i)$ under the public key pk_i in the CRS. Succinctness of party $\mathcal{P}_i$'s encoding pe_i follows by construction.
- **Evaluation for party $\mathcal{P}_i$:** On input a Boolean circuit $C\colon \{0,1\}^N \times (\{0,1\}^k)^n \to \{0,1\}$ and the public encodings $\mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n$ from the other parties, party $\mathcal{P}_i$ computes its output share as $d_i = \mathsf{PRF}(k_i, (C, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)) \in \{0,1\}$.
- **Evaluation for party $\mathcal{P}_0$:** Party $\mathcal{P}_0$ will derive its share of the output using the obfuscated program in the CRS. To ensure correctness, the obfuscated program must output the share $d_0 = C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n) \oplus d_1 \oplus \cdots \oplus d_n$. One way to do this is have the obfuscated program take as input $(C, \mathbf{x}_0, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)$ and then compute the following:
  1. Check that pe_0 is a valid commitment to $\mathbf{x}_0$. If the check fails, then the program outputs $\perp$.
  2. Decrypt each pe_i to obtain $(k_i, \mathbf{x}_i)$ and compute $d_i = \mathsf{PRF}(k_i, (C, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)) \in \{0,1\}$.
  3. Output $d_0 = C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n) \oplus d_1 \oplus \cdots \oplus d_n \in \{0,1\}$.

The basic straw-man approach achieves correctness, but does *not* satisfy succinctness. The size of the obfuscated program in the CRS scales linearly with the length of party $\mathcal{P}_0$'s input $\mathbf{x}_0$ *and* the size of the Boolean circuit C (namely, it takes $\mathbf{x}_0$ and C as input).

As we show below, we solve *both* problems using a two-party SMS scheme. In the following description, we write $(\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1)$ to denote public encodings and $(\tilde{d}_0, \tilde{d}_1)$ to denote the output shares associated with $\mathcal{P}_0$ and $\mathcal{P}_1$ in the

underlying two-party SMS scheme, respectively. As before, we follow the convention that $\mathcal{P}_0$ holds the long input (i.e., "Alice" in the earlier convention) and $\mathcal{P}_1$ holds the short input (i.e., "Bob" in the earlier convention).

Step 1: Encode the long input $\mathbf{x}_0$ using SMS. First, instead of committing to their input $\mathbf{x}_0$, party $\mathcal{P}_0$ instead uses the reusable two-party SMS protocol to construct an encoding $\widetilde{\mathsf{pe}}_0$ of her input $\mathbf{x}_0$. Party $\mathcal{P}_0$ sets their public encoding $\mathsf{pe}_0 = \widetilde{\mathsf{pe}}_0$. This is *short* by definition. With this modification, the obfuscated program no longer needs to take $\mathbf{x}_0$ as input. Instead, the obfuscated program takes $(C, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)$ as input and proceeds as follows:

1. Decrypt each pe_i to obtain $(k_i, \mathbf{x}_i)$. Then, compute an SMS encoding $\widetilde{\mathsf{pe}}_1$ of the concatenated input $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ using the underlying two-party SMS scheme (as party $\mathcal{P}_1$).[7] By design, the size of $\widetilde{\mathsf{pe}}_1$ is $\mathsf{poly}(\lambda, n, k)$.
2. Use the underlying two-party SMS protocol (as party $\mathcal{P}_1$) to evaluate the Boolean circuit $C\colon \{0,1\}^N \times (\{0,1\}^k)^n \to \{0,1\}$ on the encodings $(\mathsf{pe}_0, \widetilde{\mathsf{pe}}_1) = (\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1)$. Let $\tilde{d}_1 \in \{0,1\}$ be the output share.
3. Finally, compute $d_i = \mathsf{PRF}(k_i, (C, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)) \in \{0,1\}$ and output the blinded share $d_0' = \tilde{d}_1 \oplus d_1 \oplus \cdots \oplus d_n \in \{0,1\}$ together with the input encoding $\widetilde{\mathsf{pe}}_1$.

Returning now to the multiparty SMS protocol, the evaluation algorithm for parties $\mathcal{P}_i$ is the same as in the straw-man scheme: they simply output $d_i = \mathsf{PRF}(k_i, (C, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)) \in \{0,1\}$. To compute their share d_0, party $\mathcal{P}_0$ would proceed as follows:

1. Run the obfuscated program on input $(C, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)$ to obtain a *blinded* share $d_0' = \tilde{d}_1 \oplus d_1 \oplus \cdots \oplus d_n$ and an input encoding $\widetilde{\mathsf{pe}}_1$.
2. Use the underlying two-party SMS protocol (as party $\mathcal{P}_0$) to evaluate the circuit C on the encodings $(\mathsf{pe}_0, \widetilde{\mathsf{pe}}_1) = (\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1)$. Let $\tilde{d}_0 \in \{0,1\}$ be the output share.
3. By correctness of the underlying two-party SMS protocol, it holds that

$$\tilde{d}_0 \oplus \tilde{d}_1 = C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n).$$

   Output the share

$$\begin{aligned} d_0 = \tilde{d}_0 \oplus d_0' &= (\tilde{d}_0 \oplus \tilde{d}_1) \oplus (d_1 \oplus \cdots \oplus d_n) \\ &= C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n) \oplus d_1 \oplus \cdots \oplus d_n \in \{0,1\}. \end{aligned}$$

Thus, the shares $(d_0, d_1, \ldots, d_n)$ derived in this way constitute an additive secret sharing of $C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$. This approach removes the need to provide $\mathbf{x}_0$ as an input to the obfuscated program, but it still needs to take the circuit C as input. As such, it still does *not* satisfy succinctness.

Step 2: Encode the circuit C using SMS. To avoid having to provide the description of C to the obfuscated program, we *again* leverage SMS. Much like how we eliminated the dependence on the long input $\mathbf{x}_0$, the idea is to replace the circuit C with an SMS input encoding of C to the obfuscated program. However, in a reusable SMS protocol, the circuit C is not known at encoding time. To get around this, we will instead compose *two* separate SMS protocols: the first SMS protocol handles the input encoding (as described above), while the second handles the circuit encoding. To integrate them together, the first SMS will generate the input encodings for the second SMS instance. More concretely, we start by defining the Boolean circuit G:

1. The circuit G takes the binary representation of $(C, (\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1, \widetilde{\mathsf{td}}_1))$ as input, where $C\colon \{0,1\}^N \times (\{0,1\}^k)^n \to \{0,1\}$ is a Boolean circuit, $\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1$ are public SMS encodings for the two-party reusable SMS scheme (associated with $\mathcal{P}_0$ and $\mathcal{P}_1$), and $\widetilde{\mathsf{td}}_1$ is the state of $\mathcal{P}_1$ (which $\mathcal{P}_1$ would use to generate its shares of the output). We take C to be the long input (associated with $\mathcal{P}_0$) and $(\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1, \widetilde{\mathsf{td}}_1)$ to be the short input (associated with $\mathcal{P}_1$).

[7] Technically, the encoding procedure is a *randomized* algorithm. We derandomize this by having the program derive the randomness using a PRF evaluated on the inputs. We refer to Section 6 for the full description.

2. The circuit G computes $\mathcal{P}_1$'s output share obtained by evaluating C on the encodings $(\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1)$ (using $\widetilde{\mathsf{td}}_1$ as the state of $\mathcal{P}_1$).

In other words, the Boolean circuit G is precisely the computation described in Step 2 of the above protocol. Directly evaluating G (as done previously) would require a circuit that scales with the size of C. However, if we use a two-party *non-reusable* SMS protocol [BJSS25, AMR25] where the circuit is part of $\mathcal{P}_0$'s input, and the encoding and share-computation times of $\mathcal{P}_1$ only scale polynomially with the security parameter and the length of $\mathcal{P}_1$'s input (and polylogarithmically with the length of $\mathcal{P}_0$'s input and the output length of the circuit), then we achieve full succinctness.

Concretely, the obfuscated program in the CRS now takes as input a tuple $(\hat{C}_{\mathsf{sms}}, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)$ where $\hat{C}_{\mathsf{sms}}$ is a two-party (non-reusable) SMS encoding of the input C for the circuit G (for $\mathcal{P}_0$). The efficiency requirements for the SMS scheme ensures that $\hat{C}_{\mathsf{sms}}$ has a short description. The obfuscated program now proceeds as follows:

1. Decrypt each pe_i to obtain $(k_i, \mathbf{x}_i)$. Then, compute an SMS encoding $\widetilde{\mathsf{pe}}_1$ of the concatenated input $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ using the two-party reusable SMS scheme as party $\mathcal{P}_1$. Let $\widetilde{\mathsf{td}}_1$ be the state $\mathcal{P}_1$ would use to derive the output share.
2. Use the two-party (non-reusable) SMS protocol to compute an encoding $\widehat{\mathsf{pe}}_1$ of the input $(\mathsf{pe}_0, \widetilde{\mathsf{pe}}_1, \widetilde{\mathsf{td}}_1) = (\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1, \widetilde{\mathsf{td}}_1)$ as party $\mathcal{P}_1$.
3. Use the two-party (non-reusable) SMS protocol to compute $\mathcal{P}_1$'s output share $\hat{d}_1 \in \{0,1\}$ with respect to the input encodings $(\hat{C}_{\mathsf{sms}}, \widehat{\mathsf{pe}}_1)$.
4. Finally, compute $d_i = \mathsf{PRF}(k_i, (C, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)) \in \{0,1\}$ and output the blinded share $d'_0 = \hat{d}_1 \oplus d_1 \oplus \cdots \oplus d_n \in \{0,1\}$ together with the input encodings $\widehat{\mathsf{pe}}_1$ and $\widetilde{\mathsf{pe}}_1$.

By construction, the inputs to the obfuscated program consist of an input encoding (for $\mathcal{P}_0$) under a two-party SMS scheme and encryptions of the small inputs. Moreover, the computation performed by the obfuscated program only consists of algorithms that would be run by party $\mathcal{P}_1$ in one of the underlying two-party SMS protocols. Thus, the size of the obfuscated program scales with $\mathsf{poly}(\lambda, k, n, \log N)$, as required.[8] To compute $\mathcal{P}_0$'s share, they now proceed as follows:

1. Use the two-party (non-reusable) SMS protocol to compute an encoding $\widehat{\mathsf{pe}}_0$ of input C for the circuit G. We view $\hat{C}_{\mathsf{sms}} = \widehat{\mathsf{pe}}_0$ as the succinct encoding of the circuit C.
2. Run the obfuscated program on input $(\hat{C}_{\mathsf{sms}}, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)$ to obtain a blinded bit $d'_0 = \hat{d}_1 \oplus d_1 \oplus \cdots \oplus d_n$ along with input encodings $\widehat{\mathsf{pe}}_1$ and $\widetilde{\mathsf{pe}}_1$.
3. Run the underlying two-party non-reusable SMS protocol on the encodings $(\widehat{\mathsf{pe}}_0, \widehat{\mathsf{pe}}_1)$. Let $\hat{d}_0$ be the output share. By correctness of the two-party SMS protocol, we have
$$\hat{d}_0 \oplus \hat{d}_1 = G(C, (\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1, \widetilde{\mathsf{td}}_1)) = \tilde{d}_1,$$
where $\tilde{d}_1$ is $\mathcal{P}_1$'s share obtained by evaluating C on encodings $(\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1)$ in the underlying two-party reusable SMS protocol.
4. Run the underlying two-party reusable SMS protocol on the encodings $(\mathsf{pe}_0, \widetilde{\mathsf{pe}}_1) = (\widetilde{\mathsf{pe}}_0, \widetilde{\mathsf{pe}}_1)$ with the circuit C to obtain an output share $\tilde{d}_0$. By correctness of the underlying SMS protocol, we have
$$\tilde{d}_0 \oplus \tilde{d}_1 = C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n).$$

[8] When considering Boolean circuits with m output bits, we incur an additional dependence on $\log m$. The extra $\log m$ factor comes from the fact that the underlying two-party SMS schemes with single-bit output has negligible correctness error, so extending to multiple output bits requires taking a union bound over the number of output bits. Note that in our setting, we can always bound m (as well as N) by 2^λ, so all $\mathsf{poly}(\log N, \log m)$ terms can be absorbed by a $\mathsf{poly}(\lambda)$ term.

5. Output the share

$$\begin{aligned} d_0 &= \tilde{d}_0 \oplus \hat{d}_0 \oplus d'_0 \\ &= \tilde{d}_0 \oplus (\hat{d}_0 \oplus \hat{d}_1) \oplus (d_1 \oplus \cdots \oplus d_n) \\ &= (\tilde{d}_0 \oplus \tilde{d}_1) \oplus (d_1 \oplus \cdots \oplus d_n) \\ &= C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n) \oplus d_1 \oplus \cdots \oplus d_n, \end{aligned}$$

and correctness follows.

Proving security. We now provide a brief overview of the main ideas underlying the security analysis. Recall that our objective is to show that each party's input encoding pe_i hides their input $\mathbf{x}_i$. First, consider $\mathcal{P}_0$. By construction, party $\mathcal{P}_0$'s encoding is an encoding of $\mathbf{x}_0$ using an underlying two-party SMS protocol. Thus, security for $\mathcal{P}_0$ follows immediately by security of the underlying two-party SMS protocol.

Security for $\mathcal{P}_i$ (for $i \in [n]$) is more involved. Recall first that $\mathcal{P}_i$'s encoding is an encryption of $(k_i, \mathbf{x}_i)$ under the public key pk_i in the CRS. Moreover, the corresponding decryption key sk_i is hardwired inside the obfuscated program in the CRS. Thus, we cannot directly invoke semantic security of the underlying public-key encryption scheme to argue that $\mathcal{P}_i$'s encoding hides their input $\mathbf{x}_i$. Instead, we proceed in a sequence of hybrid experiments where in the final distribution, the CRS as well as party $\mathcal{P}_i$'s input encoding are *independent* of their input $\mathbf{x}_i$:

- In the first sequence of hybrids, we replace party $\mathcal{P}_i$'s encoding pe_i (i.e., the encryption of $(k_i, \mathbf{x}_i)$) with an encryption of $\bot$. Note that to ensure functional equivalence for the program in the CRS, we also need to hardcode the value $(k_i, \mathbf{x}_i)$ into the obfuscated program as the "decryption" of pe_i.

  To argue this, we adopt a Naor-Yung approach [NY90]. Namely, we modify the scheme to include two independent public keys $\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}$ for each party $i \in [n]$ in the CRS. Then, we have party $\mathcal{P}_i$ encrypt the pair $(k_i, \mathbf{x}_i)$ with respect to $\mathsf{pk}_{i,0}$ *and* $\mathsf{pk}_{i,1}$. Their public encoding now includes a pair of ciphertexts $\mathsf{ct}_{i,0}$ and $\mathsf{ct}_{i,1}$. In addition, $\mathcal{P}_i$ also includes a (simulation-sound) non-interactive zero-knowledge (NIZK) proof [Sah99, DDO+01] that the two ciphertexts $\mathsf{ct}_{i,0}$ and $\mathsf{ct}_{i,1}$ encrypt the *same* value. The key insight underlying the Naor-Yung approach is that we can decrypt $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1})$ with knowledge of either the secret key $\mathsf{sk}_{i,0}$ associated with $\mathsf{pk}_{i,0}$ or the secret key $\mathsf{sk}_{i,1}$ associated with $\mathsf{pk}_{i,1}$. Thus, we can consider a reduction where the obfuscated program uses $\mathsf{sk}_{i,0}$ to decrypt $\mathcal{P}_i$'s ciphertext while still being able to invoke semantic security for ciphertexts encrypted with respect to $\mathsf{pk}_{i,1}$. Then, we switch the obfuscated program to use $\mathsf{sk}_{i,1}$ to decrypt, which allows us to invoke semantic security with respect to $\mathsf{pk}_{i,0}$. This allows us to replace the ciphertexts in $\mathcal{P}_i$'s encoding with an encryption of a value that is independent of $(k_i, \mathbf{x}_i)$. A similar Naor-Yung approach also featured in the construction of functional encryption from $i\mathcal{O}$ [GGH+13].

- After the previous step, party $\mathcal{P}_i$'s public encoding pe_i no longer depends on $\mathbf{x}_i$, but at the same time, we have also hardcoded $\mathcal{P}_i$'s input $\mathbf{x}_i$ into the obfuscated program itself (i.e., the decryption of pe_i is always fixed to be $(k_i, \mathbf{x}_i)$). In the next sequence of hybrids, our goal is to erase the input $\mathbf{x}_i$ from the description of the obfuscated program. To do so, we consider every possible choice of public encodings $(\mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)$, and for each possible tuple, we argue that by security of the underlying two-party SMS protocols (as well as of the PRF), we can replace the decryption of pe_i with $\bot$ in the obfuscated program.

  Since this step requires iterating over all possible inputs $(\mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n)$, it is critical that the public encodings are short (polylogarithmic in the length of the long input $\mathbf{x}_0$). In this case, as long as the underlying two-party SMS protocols, the indistinguishability obfuscation scheme, and the PRF are sub-exponentially secure, security holds.

We refer to Section 6 for the full details of the construction and security analysis.

3 Preliminaries

Throughout this work, we write λ to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n] := \{1, \ldots, n\}$. For integers $a < b \in \mathbb{Z}$, we write $[a, b] := \{a, a+1, \ldots, b\}$. We write $\mathsf{poly}(\lambda)$ to denote a function that is bounded by a fixed polynomial in λ, and $\mathsf{negl}(\lambda)$ to denote a negligible function (i.e., a function which is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$). We say a sequence of events (indexed by a security parameter λ) occurs with overwhelming probability if the event occurs with probability $1 - \mathsf{negl}(\lambda)$. We say a cryptographic primitive has sub-exponential security if there exists a constant $0 < \varepsilon < 1$ such that for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the probability that $\mathcal{A}$ can break the security of this primitive is $2^{-\lambda^{\varepsilon}} \cdot \mathsf{negl}(\lambda)$.

Local rounding. Take $p, q \in \mathbb{N}$ where $q > p$. Define the function $\mathsf{Round}_{q \to p} \colon \mathbb{Z}_q \to \mathbb{Z}_p$ that takes an input $t \in \mathbb{Z}_q$, lifts it to the integers (over the interval $(-q/2, q/2]$) and outputs $\lfloor (p/q) \cdot t \rceil$. We extend the rounding operation $\mathsf{Round}_{q \to p}$ to operate on vectors and matrices component-wise. We now recall the local rounding lemma from [DHRW16, BKS19]:

Lemma 3.1 (Local Rounding [BKS19, adapted])**.** *Take any $p, q \in \mathbb{N}$ where $q > p$. Then, for all $e \in \mathbb{Z}$,*

$$\Pr[\mathsf{Round}_{q \to p}(t) = \mathsf{Round}_{q \to p}(t + e) : t \xleftarrow{\mathrm{R}} \mathbb{Z}_q] \geq 1 - O\left(\frac{p|e|}{q}\right).$$

Next, we define the main cryptographic primitives we use in this work.

Leveled homomorphic encryption with nearly-linear decryption. Next, we recall the definition of a (leveled) homomorphic encryption scheme. In our applications, we additionally require that the decryption algorithm is a *nearly linear* function of the secret key and the ciphertext. The nearly-linear decryption property is satisfied by most lattice-based homomorphic encryption schemes (e.g., [BV11, Bra12, BGV12, GSW13]).

Definition 3.2 (Leveled Homomorphic Encryption)**.** Let λ be a security parameter and d be a depth parameter. Let $n = n(\lambda, d)$ be the dimension of the secret and $p = p(\lambda, d)$ be the modulus. A (secret-key) leveled homomorphic encryption scheme for Boolean circuits with nearly-linear decryption consists of a tuple of efficient algorithms $\Pi_{\mathsf{LHE}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ with the following syntax:

- $\mathsf{KeyGen}(1^{\lambda}, 1^{d}) \to (\mathsf{pk}, \mathbf{s})$: On input the security parameter λ and the depth bound d, the key-generation algorithm outputs a public evaluation key pk and a secret key $\mathbf{s} \in \mathbb{Z}_p^n$.
- $\mathsf{Enc}(\mathbf{s}, \mu) \to \mathsf{ct}$: On input the secret key $\mathbf{s} \in \mathbb{Z}_p^n$ and a message $\mu \in \{0, 1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Eval}(\mathsf{pk}, C, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)) \to \mathbf{c}$: On input the public evaluation key pk, a Boolean circuit $C \colon \{0,1\}^\ell \to \{0,1\}$, and a tuple of ℓ ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the evaluation algorithm outputs an evaluated ciphertext $\mathbf{c} \in \mathbb{Z}_p^n$. This algorithm is deterministic.

We require Π_{LHE} satisfy the following properties:

- **Nearly-linear decryption:** For all $\lambda, d \in \mathbb{N}$, all Boolean circuits $C \colon \{0,1\}^\ell \to \{0,1\}$ of depth at most d, all inputs $\mathbf{x} \in \{0,1\}^\ell$, we have that

$$\Pr\left[\mathbf{s}^{\mathsf{T}}\mathbf{c} = C(\mathbf{x}) \cdot \lfloor p/2 \rfloor + e \text{ where } |e| < \sqrt{p} : \begin{array}{c} (\mathsf{pk}, \mathbf{s}) \leftarrow \mathsf{KeyGen}(1^{\lambda}, 1^{d}) \\ \forall i \in [\ell] : \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathbf{s}, x_i) \\ \mathbf{c} \leftarrow \mathsf{Eval}(\mathsf{pk}, C, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)) \end{array}\right] = 1.$$

- **CPA-security:** For a security parameter λ, a depth parameter $d \in \mathbb{N}$, and a bit $b \in \{0,1\}$, we define the CPA-security game for an adversary $\mathcal{A}$ as follows:
  - The challenger samples $(\mathsf{pk}, \mathbf{s}) \leftarrow \mathsf{KeyGen}(1^{\lambda}, 1^{d})$ and gives $(1^{\lambda}, 1^{d}, \mathsf{pk})$ to the adversary $\mathcal{A}$.
  - The adversary $\mathcal{A}$ can now make encryption queries on pairs of messages (μ_0, μ_1). On each query, the challenger responds with $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathbf{s}, \mu_b)$.

– At the end of the experiment, the adversary outputs a bit $b' \in \{0,1\}$ which is the output of the experiment.

The scheme is CPA-secure if for all polynomials $d = d(\lambda)$ and all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda).$$

Remark 3.3 (Supporting Multi-Bit Outputs)**.** We can extend homomorphic evaluation to circuits with multi-bit outputs by repeatedly applying Eval to compute each output bit individually. Specifically, for a circuit $C\colon \{0,1\}^\ell \to \{0,1\}^k$, we evaluate the circuits $C_1, \ldots, C_k$ where C_i is the circuit that computes the i^{th} bit of the output of C.

Theorem 3.4 (Leveled Homomorphic Encryption [BV11, Bra12, BGV12, GSW13])**.** *Let λ be the security parameter and d be the depth parameter. Under the learning with errors (LWE) assumption with a sub-exponential modulus-to-noise ratio, for every sufficiently large $n = \mathsf{poly}(\lambda, d)$, there exists a leveled homomorphic encryption scheme for Boolean circuits with nearly-linear decryption and secret keys of dimension n and modulus $p = 2^{\tilde{O}(d)}$.*

Algebraic homomorphic MAC. In this work, we also use the notion of a (leveled) algebraic homomorphic MAC introduced in the work of [ILL25a]. We use a slightly different syntax where the encoding keys $\mathbf{k}$ associated with input bits are arbitrary strings and not generated by a separate Auth function. We give our definition below and refer to Remark 3.7 for a comparison with the syntax from [ILL25a].

Definition 3.5 (Leveled Algebraic Homomorphic MAC [ILL25a, adapted])**.** Let λ be a security parameter, $n = n(\lambda)$ be a dimension, and $p = p(\lambda)$ be a modulus. A leveled algebraic homomorphic MAC over $\mathbb{Z}_p^n$ is a tuple of efficient algorithms $\Pi_{\mathsf{AHMAC}} = (\mathsf{Setup}, \mathsf{EvalKey}, \mathsf{EvalTag})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^d, \mathbf{s}) \to (\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk})$: On input the security parameter 1^λ, a depth bound 1^d, and the global offset $\mathbf{s} \in \mathbb{Z}_p^n$, the setup algorithm outputs a secret input MAC key $\mathbf{s}_{\mathsf{in}} \in \mathbb{Z}_p^n$, a secret key sk, and an evaluation key evk.
- $\mathsf{EvalKey}(\mathsf{sk}, \mathsf{evk}, C, (\mathbf{k}_1, \ldots, \mathbf{k}_\ell)) \to \mathbf{k}_C$: On input the secret key sk, the evaluation key evk, a Boolean circuit $C\colon \{0,1\}^\ell \to \{0,1\}$, and a vector of keys $\mathbf{k}_1, \ldots, \mathbf{k}_\ell \in \mathbb{Z}_p^n$ associated with the inputs to C, the key-evaluation function outputs a key $\mathbf{k}_C \in \mathbb{Z}_p^n$ associated with the output of C.
- $\mathsf{EvalTag}(\mathsf{evk}, C, \mathbf{x}, \boldsymbol{\sigma}_{\mathbf{x}}) \to \boldsymbol{\sigma}'$: On input the evaluation key evk, a Boolean circuit $C\colon \{0,1\}^\ell \to \{0,1\}$, an input $\mathbf{x} \in \{0,1\}^\ell$ and their respective tags $\boldsymbol{\sigma}_{\mathbf{x}} = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_\ell)$, the tag-evaluation function outputs a tag $\boldsymbol{\sigma}' \in \mathbb{Z}_p^n$.

We note that EvalKey and EvalTag are deterministic algorithms. We require Π_{AHMAC} satisfy the following properties:

- **Correctness:** For all polynomials $d = d(\lambda)$ and $s = s(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all global offsets $\mathbf{s} \in \mathbb{Z}_p^n$, all Boolean circuits $C\colon \{0,1\}^\ell \to \{0,1\}$ of size at most $s(\lambda)$ and depth at most $d(\lambda)$, all inputs $\mathbf{x} \in \{0,1\}^\ell$, and all keys $\mathbf{k}_1, \ldots, \mathbf{k}_\ell \in \mathbb{Z}_p^n$,

$$\Pr\left[\boldsymbol{\sigma}' = C(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_C \;:\; \begin{array}{c}(\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^d, \mathbf{s}) \\ \forall i \in [\ell] : \boldsymbol{\sigma}_i = x_i \cdot \mathbf{s}_{\mathsf{in}} + \mathbf{k}_i \\ \mathbf{k}_C = \mathsf{EvalKey}(\mathsf{sk}, \mathsf{evk}, C, (\mathbf{k}_1, \ldots, \mathbf{k}_\ell)) \\ \boldsymbol{\sigma}' = \mathsf{EvalTag}(\mathsf{evk}, C, \mathbf{x}, (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_\ell))\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

- **Adaptive security:** For a bit $b \in \{0,1\}$, an adversary $\mathcal{A}$, and a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, we define the adaptive security experiment as follows:
  - On input the security parameter 1^λ, algorithm $\mathcal{A}$ outputs the depth bound 1^d and the global secret $\mathbf{s} \in \mathbb{Z}_p^n$. The challenger computes evk as follows:
    - If $b = 0$, the challenger samples $(\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^d, \mathbf{s})$.
    - If $b = 1$, the challenger samples $(\mathsf{evk}, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d)$.

The challenger gives evk to $\mathcal{A}$.

- Algorithm $\mathcal{A}$ can now make adaptive queries on inputs $\mathbf{x} \in \{0,1\}^\ell$. The challenger responds as follows:
  - If $b = 0$, the challenger samples $\mathbf{k} \xleftarrow{R} (\mathbb{Z}_p^n)^\ell$ and responds with $\boldsymbol{\sigma} \leftarrow \mathbf{x} \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}$.
  - If $b = 1$, the challenger responds with $(\boldsymbol{\sigma}, \mathsf{st}) \leftarrow \mathcal{S}_2(\mathsf{st}, 1^\ell)$.

  The challenger gives $\boldsymbol{\sigma}$ to $\mathcal{A}$.
- At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

We say that Π_{AHMAC} satisfies adaptive security if there exists an efficient simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda)$$

in the adaptive security game defined above. We say Π_{AHMAC} is selectively secure if the adversary has to commit to all of its queries at the beginning of the security game (before seeing evk).

- **Succinctness:** There exists a universal polynomial $\mathsf{poly}(\cdot,\cdot)$ such that for all $\lambda, d \in \mathbb{N}$, all $\mathbf{s} \in \mathbb{Z}_p^n$, the evaluation key evk output by $\mathsf{Setup}(1^\lambda, 1^d, \mathbf{s})$ has size at most $\mathsf{poly}(\lambda, d)$.

Remark 3.6 (Arithmetic Computations)**.** It is straightforward to extend the class of functions supported by the algebraic MAC scheme to support functions $f\colon \{0,1\}^\ell \to \mathbb{Z}_p$ that output $\mathbb{Z}_p$ elements instead of bits. To illustrate this, let $N = \lceil \log p \rceil$ and let $C_1, \ldots, C_N$ be Boolean circuits where $C_j(\mathbf{x})$ computes the j^{th} least-significant-bit of $f(\mathbf{x})$. This means we can write

$$f(\mathbf{x}) = \sum_{j \in [N]} 2^{j-1} \cdot C_j(\mathbf{x}) \in \mathbb{Z}_p.$$

We can now define EvalKey and EvalTag for functions f as follows:

- $\mathsf{EvalKey}(\mathsf{sk}, \mathsf{evk}, f, (\mathbf{k}_1, \ldots, \mathbf{k}_\ell)) \to \mathbf{k}_f$: For each $j \in [N]$, compute $\mathbf{k}_j = \mathsf{EvalKey}(\mathsf{sk}, \mathsf{evk}, C_j, (\mathbf{k}_1, \ldots, \mathbf{k}_\ell))$. Output $\mathbf{k}_f = \sum_{j \in [N]} 2^{j-1} \cdot \mathbf{k}_j \in \mathbb{Z}_p^n$.
- $\mathsf{EvalTag}(\mathsf{evk}, f, \mathbf{x}, \boldsymbol{\sigma}_{\mathbf{x}})$: For each $j \in [N]$, compute $\boldsymbol{\sigma}'_j = \mathsf{EvalTag}(\mathsf{evk}, C_j, \mathbf{x}, \boldsymbol{\sigma}_{\mathbf{x}})$ and output $\boldsymbol{\sigma}'_f = \sum_{j \in [N]} 2^{j-1} \cdot \boldsymbol{\sigma}'_j \in \mathbb{Z}_p^n$.

Correctness now follows by linearity. Namely if for all $j \in [N]$, we have that

$$\boldsymbol{\sigma}'_j = C_j(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_j,$$

then,

$$\boldsymbol{\sigma}'_f = \sum_{j \in [N]} 2^{j-1} \cdot \boldsymbol{\sigma}'_j = \sum_{j \in [N]} 2^{j-1} \cdot (C_j(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_j) = f(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_f \in \mathbb{Z}_p^n.$$

This naturally extends to support computing vector-valued functions $f\colon \{0,1\}^\ell \to \mathbb{Z}_p^t$ where $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_t(\mathbf{x}))$ where $f_i\colon \{0,1\}^\ell \to \mathbb{Z}_p$ by separately evaluating each component. In this case, the algebraic homomorphic MAC invariant becomes

$$\boldsymbol{\sigma}'_f = f(\mathbf{x}) \otimes \mathbf{s} + \mathbf{k}_f \in \mathbb{Z}_p^{nt},$$

where $\boldsymbol{\sigma}'_f \in \mathbb{Z}_p^{nt}$ is the vertical concatenation of $\boldsymbol{\sigma}'_{f_1}, \ldots, \boldsymbol{\sigma}'_{f_t} \in \mathbb{Z}_p^n$ and $\mathbf{k}_f \in \mathbb{Z}_p^{nt}$ is the vertical concatenation of $\mathbf{k}_{f_1}, \ldots, \mathbf{k}_{f_t} \in \mathbb{Z}_p^n$.

Remark 3.7 (Comparison with [ILL25a])**.** We also briefly compare our algebraic homomorphic MAC syntax with that from [ILL25a]. There, the authors have a separate Auth function that takes as input the secret key, an input bit $x \in \{0,1\}$ and a tag id, and then outputs a MAC $\boldsymbol{\sigma}$ on x with respect to the tag. The tag can be viewed as an index or a label that is used to derive the encoding key $\mathbf{k}$ associated with $\boldsymbol{\sigma}$. In our setting, we do not demand a separate encoding algorithm and instead consider correctness with respect to an arbitrary set of input encoding keys, and security in the setting where the input encoding keys are uniformly random. Our syntax is more convenient for our main application to reusable 2-party SMS.

In Section 7, we show how to construct a leveled algebraic homomorphic MAC from the plain LWE assumption with a sub-exponential modulus-to-noise ratio. Our construction can be viewed as an adaptation of the leveled algebraic homomorphic MAC from [ILL25a] based on ring LWE to the setting of integer lattices. We summarize the instantiation we use here, and give the formal proof of the statement in Section 7.

Theorem 3.8 (Leveled Algebraic MAC from LWE)**.** *Let λ be a security parameter and d be the depth parameter. Under the LWE assumption with a $2^{\tilde{O}(n^\varepsilon)}$-modulus-to-noise ratio (where $0 < \varepsilon < 1$ is a constant), for every sufficiently large $n = \mathsf{poly}(\lambda, d)$ and every $\lambda^{\omega(1)} \le p \le 2^{\tilde{O}(n^\varepsilon)}$, there exists a leveled algebraic homomorphic MAC scheme over $\mathbb{Z}_p^n$ where the size of the evaluation key is $|\mathsf{evk}| = \mathsf{poly}(\lambda, d)$.*

Simultaneous-message and succinct (SMS) secure computation. We recall the notion of simultaneous-message and succinct (SMS) secure computation from [BJSS25]. An SMS scheme is a two-party protocol between a party $\mathcal{P}_0$ who holds a *long* private input $\mathbf{x}_0 \in \{0,1\}^N$ and a party $\mathcal{P}_1$ who holds a *short* private input $\mathbf{x}_1 \in \{0,1\}^k$. Their goal is to compute an additive secret sharing of a *fixed* function $C(\mathbf{x}_0, \mathbf{x}_1)$ where $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}^m$. The parties have access to a common reference string (CRS). The protocol consists of one message from $\mathcal{P}_0$ to $\mathcal{P}_1$ and one message from $\mathcal{P}_1$ to $\mathcal{P}_0$. Moreover, each party's message is a function of only the CRS and their local input (i.e., neither party's message depends on the other party's message, and as such, they can be sent "simultaneously"). The security requirement is that each party's message should computationally hide their input while the efficiency requirement is that the length of the CRS and the length of each party's message should only scale polylogarithmically with the description of the function C and the length N of the long input $\mathbf{x}_0$. We give the formal syntax below:

Definition 3.9 (Simultaneous-Message and Succinct Secure Computation [BJSS25])**.** Let λ be a security parameter and let $C = \{C_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of bounded-depth Boolean circuits indexed by a parameter κ. We assume that each circuit $C \in C_\kappa$ is a Boolean circuit of depth at most $d = d(\kappa)$ and of type $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}^m$, where $d = d(\kappa)$, $N = N(\kappa)$, $k = k(\kappa)$, and $m = m(\kappa)$ are polynomially-bounded. An SMS scheme Π_{SMS} for C is a tuple of efficient algorithms $\Pi_{\mathsf{SMS}} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Decode})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^\kappa) \to \mathsf{crs}$: On input the security parameter λ and the circuit-family parameter κ, the setup algorithm outputs a common reference string crs. We assume that crs implicitly includes a description of the circuit-family parameter κ.
- $\mathsf{Encode}(\mathsf{crs}, i, \mathsf{inp}) \to (\mathsf{pe}_i, \mathsf{td}_i)$: On input the common reference string crs, a party index $i \in \{0,1\}$, and an input inp, the encoding algorithm outputs a public encoding pe_i and a trapdoor td_i. If $i = 0$, we interpret $\mathsf{inp} = (\mathbf{x}_0, C)$ where $\mathbf{x}_0 \in \{0,1\}^N$ and $C \in C_\kappa$. If $i = 1$, we view $\mathsf{inp} = \mathbf{x}_1 \in \{0,1\}^k$.
- $\mathsf{Decode}(1^m, \mathsf{crs}, i, \mathsf{pe}_{1-i}, \mathsf{td}_i) \to d_i$: On input the output length m, the common reference string crs, a party index $i \in \{0,1\}$, the public encoding pe_{1-i} of the other party, and the trapdoor td_i, the decoding algorithm outputs the decoding information d_i.

We want this scheme to satisfy the following properties:

- **Correctness:** For all polynomials $s = s(\lambda) \le 2^\lambda$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all $\kappa \in \mathbb{N}$, all inputs $\mathbf{x}_0 \in \{0,1\}^N$, $\mathbf{x}_1 \in \{0,1\}^k$ and all circuits $C \in C_\kappa$ where $|C| \le s(\lambda)$, we have:

$$\Pr\left[d_0 \oplus d_1 = C(\mathbf{x}_0, \mathbf{x}_1) : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa) \\ (\mathsf{pe}_0, \mathsf{td}_0) \leftarrow \mathsf{Encode}(\mathsf{crs}, 0, (\mathbf{x}_0, C)) \\ (\mathsf{pe}_1, \mathsf{td}_1) \leftarrow \mathsf{Encode}(\mathsf{crs}, 1, \mathbf{x}_1) \\ \forall i \in \{0,1\} : d_i \leftarrow \mathsf{Decode}(1^m, \mathsf{crs}, i, \mathsf{pe}_{1-i}, \mathsf{td}_i) \end{array} \right] \ge 1 - \mathsf{negl}(\lambda).$$

- **Security:** For a bit $b \in \{0,1\}$ and a party index $i \in \{0,1\}$, we define the security game between an adversary $\mathcal{A}$ and a challenger as follows:
  - On input the security parameter 1^λ and the party index $i \in \{0,1\}$, the adversary outputs the circuit-family parameter 1^κ.

- The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ and replies to $\mathcal{A}$ with crs.
- The adversary outputs two messages $\mathsf{inp}^{(0)}$ and $\mathsf{inp}^{(1)}$ where
  - $\mathsf{inp}^{(0)} = (\mathbf{x}^{(0)}, C^{(0)}), \mathsf{inp}^{(1)} = (\mathbf{x}^{(1)}, C^{(1)})$, $\mathbf{x}^{(0)}, \mathbf{x}^{(1)} \in \{0,1\}^N$, and $C^{(0)}, C^{(1)} \in \mathcal{C}_\kappa$ such that $|C^{(0)}| = |C^{(1)}|$ if $i = 0$.
  - $\mathsf{inp}^{(0)}, \mathsf{inp}^{(1)} \in \{0,1\}^k$ if $i = 1$.
- The challenger computes $(\mathsf{pe}_i, \mathsf{td}_i) \leftarrow \mathsf{Encode}(\mathsf{crs}, i, \mathsf{inp}^{(b)})$ and gives pe_i to $\mathcal{A}$.
- Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

We say that Π_{SMS} is *adaptively* secure if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and all party indices $i \in \{0,1\}$, we have that

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda).$$

We say that Π_{SMS} is *selectively* secure if the adversary has to commit to $(\mathsf{inp}^{(0)}, \mathsf{inp}^{(1)})$ at the beginning of the security game before it receives the CRS.

- **Succinctness:** We say that the scheme satisfies succinctness if there exists a polynomial $\mathsf{poly}(\cdot)$ such that the size of the crs output by $\mathsf{Setup}(1^\lambda, 1^\kappa)$ is $\mathsf{poly}(\lambda, d, k)$ and the size of the public encodings pe_0 and pe_1 output by $\mathsf{Encode}(\mathsf{crs}, 0, \cdot)$ and $\mathsf{Encode}(\mathsf{crs}, 1, \cdot)$ satisfy $|\mathsf{pe}_0| = \mathsf{poly}(\lambda, d)$ and $|\mathsf{pe}_1| = \mathsf{poly}(\lambda, d, k)$. Recall that $d = d(\kappa)$ and $k = k(\kappa)$ are polynomial functions of κ.[9]

Theorem 3.10 (Simultaneous-Message and Succinct Secure Computation [BJSS25, AMR25])**.** *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ be any family of bounded-depth Boolean circuits. Under the LWE assumption with a sub-exponential modulus-to-noise ratio, there exists a simultaneous-message and succinct secure computation scheme for $\mathcal{C}$. Furthermore, assuming the sub-exponential hardness of LWE, there exists an SMS scheme for $\mathcal{C}$ with sub-exponential hardness.*

4 Defining Reusable Multiparty SMS

In this section, we introduce our notion of reusable multiparty simultaneous-message and succinct secure computation protocols. Consider a setting with n clients $\mathcal{P}_1, \ldots, \mathcal{P}_n$ along with a server $\mathcal{P}_0$. Each client $\mathcal{P}_i$ has a (private) input $\mathbf{x}_i \in \{0,1\}^k$ and the server has a (private) input $\mathbf{x}_0 \in \{0,1\}^N$. We typically think of N and m as being much larger than k. A reusable multiparty SMS scheme is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Eval})$ with the following properties:

- The Setup algorithm takes in the security parameter λ and outputs the common reference string crs.
- Each party $\mathcal{P}_i$ can apply the Encode algorithm on their input $\mathbf{x}_i$ to obtain a public encoding pe_i associated with their input and a secret decoding trapdoor td_i.
- Finally, each party $\mathcal{P}_i$ can apply the evaluation algorithm Eval on a Boolean circuit C, the public encodings for all of the parties $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$, and their decoding trapdoor td_i to obtain a share of the output d_i.

The correctness requirement is that the parties' shares satisfy $d_0 \oplus d_1 \oplus \cdots \oplus d_n = C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$. For security, we require that each party's public encoding pe_i computationally hide their input $\mathbf{x}_i$. Finally, the succinctness property requires that the size of the public encoding pe and the size of the crs scale polylogarithmically with the length of the server's input N and the length of the output m. Note that since we only consider correctness in settings where $N, m \leq 2^\lambda$, the polylogarithmic terms can be absorbed into a $\mathsf{poly}(\lambda)$ term without loss of generality.

There are two main differences between SMS and reusable multiparty SMS protocols. First, we extend the SMS notion to multiple parties. Second, in the case of SMS, the public encoding generated by $\mathcal{P}_0$ depends on the circuit being evaluated. In contrast, with reusable SMS, the circuit is provided as input to the Eval algorithm. This means the same set of public encodings could be used to evaluate different circuits. This key property makes the public encodings

[9] Since we only demand correctness to hold for circuits with size at most 2^λ, and the size of the circuit is an upper bound on the length of the large input N and the length of the output m, we can absorb any terms that are polylogarithmic in N or m into the security parameter λ.

reusable in our setting, whereas the public encodings in the existing SMS notion are not reusable. On the flip side, in a reusable SMS, the amount of computation each party $\mathcal{P}_i$ has to perform to compute their output share d_i can depend on the size of the Boolean circuit C (since we allow the evaluation algorithm to take C as input). In contrast, in vanilla SMS, only the server $\mathcal{P}_0$ has to perform work proportional to the size of C. The remaining parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ only need to perform work proportional to the length of their (short) input and the length of the output to compute their output share d_i. Note that our reusable multiparty SMS protocol based on $i\mathcal{O}$ (Section 6) has the property that $\mathcal{P}_i$'s work scales only with the output length of C (and *not* the size of C), much like in vanilla multiparty SMS.

Definition 4.1 (Reusable Multiparty SMS)**.** Let λ be a security parameter and let $\mathcal{C} = \{C_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of bounded-depth Boolean circuits indexed by a parameter κ. We assume that each circuit $C \in \mathcal{C}_\kappa$ is a Boolean circuit of depth at most $d = d(\kappa)$ and of type $C\colon \{0,1\}^N \times (\{0,1\}^k)^n \to \{0,1\}^m$, where $d = d(\kappa)$, $N = N(\kappa)$, $k = k(\kappa)$, $n = n(\kappa)$, and $m = m(\kappa)$ are polynomially-bounded. A reusable multiparty SMS protocol $\Pi_{\mathsf{RM\text{-}SMS}}$ for $\mathcal{C}$ is a tuple of efficient algorithms $\Pi_{\mathsf{RM\text{-}SMS}} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Eval})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^\kappa) \to \mathsf{crs}$: On input the security parameter λ and the circuit-family parameter κ, the setup algorithm outputs a common reference string crs. We assume that crs implicitly contains the circuit-family parameter 1^κ.
- $\mathsf{Encode}(\mathsf{crs}, i, \mathbf{x}_i) \to (\mathsf{pe}_i, \mathsf{td}_i)$: On input the common reference string crs, a party index $i \in [0,n]$, and their input $\mathbf{x}_i$ (where $\mathbf{x}_i \in \{0,1\}^N$ if $i = 0$ and $\mathbf{x}_i \in \{0,1\}^k$ if $i \in [n]$), the encoding algorithm outputs a public encoding pe_i and a trapdoor td_i.
- $\mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n), C, i, \mathsf{td}) \to d_i$: On input the common reference string crs, a collection of public encodings $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$, a circuit $C \in \mathcal{C}_\kappa$, a party index $i \in [0,n]$, and their trapdoor td, the evaluation algorithm outputs an output share d_i.

We want this scheme to satisfy the following properties:

- **Correctness:** For all polynomials $s = s(\lambda) \le 2^\lambda$ and $t = t(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all $\kappa \in \mathbb{N}$, all inputs $\mathbf{x}_0 \in \{0,1\}^N, \mathbf{x}_1, \ldots, \mathbf{x}_n \in \{0,1\}^k$, and all circuits $C_1, \ldots, C_t \in \mathcal{C}_\kappa$ where $|C_i| \le s(\lambda)$, we have

$$\Pr\left[\forall j \in [t], \hat{d}_j = C_j(\mathbf{x}_0, \ldots, \mathbf{x}_n) : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa) \\ \forall i \in [0,n] : (\mathsf{pe}_i, \mathsf{td}_i) \leftarrow \mathsf{Encode}(\mathsf{crs}, i, \mathbf{x}_i) \\ \forall i \in [0,n], j \in [t] : d_{i,j} \leftarrow \mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n), C_j, i, \mathsf{td}_i) \\ \hat{d}_j = d_{0,j} \oplus d_{1,j} \oplus \cdots \oplus d_{n,j} \end{array}\right] \ge 1 - \mathsf{negl}(\lambda).$$

- **Security:** For a bit $b \in \{0,1\}$ and a party index $i \in [0,n]$, we define the security game between an adversary $\mathcal{A}$ and a challenger as follows:
  - On input the security parameter 1^λ and the party index $i \in [0,n]$, the adversary $\mathcal{A}$ outputs the circuit-family parameter 1^κ.
  - The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ and gives crs to $\mathcal{A}$.
  - Algorithm $\mathcal{A}$ outputs two messages $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}$ where $\mathbf{x}^{(0)}, \mathbf{x}^{(1)} \in \{0,1\}^N$ if $i = 0$ and $\mathbf{x}^{(0)}, \mathbf{x}^{(1)} \in \{0,1\}^k$ if $i \in [n]$.
  - The challenger computes $(\mathsf{pe}_i, \mathsf{td}_i) \leftarrow \mathsf{Encode}(\mathsf{crs}, i, \mathbf{x}^{(b)})$ and responds to $\mathcal{A}$ with pe_i.
  - Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  We say that $\Pi_{\mathsf{RM\text{-}SMS}}$ is *adaptively* secure if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all party indices $i \in [0,n]$

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda)$$

  in the above security game. We say that $\Pi_{\mathsf{RM\text{-}SMS}}$ is *selectively* secure if the adversary $\mathcal{A}$ in the security game has to commit to the input $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ at the beginning of the security game *before* seeing the CRS.

- **Succinctness:** We say that the scheme satisfies succinctness if there exists a polynomial $\mathsf{poly}(\cdot)$ such that the size of the crs output by $\mathsf{Setup}(1^\lambda, 1^\kappa)$ satisfies $|\mathsf{crs}| = \mathsf{poly}(\lambda, d, n, k)$, and the size of the public encodings pe_i output by $\mathsf{Encode}(\mathsf{crs}, i, \cdot)$ satisfies $|\mathsf{pe}_0| = \mathsf{poly}(\lambda, d)$ and for every $i \in [n]$, we have $|\mathsf{pe}_i| = \mathsf{poly}(\lambda, d, k)$.

Remark 4.2 (Supporting Multi-Bit Outputs)**.** We note that reusable SMS for circuits with single-bit outputs implies reusable SMS for circuits with multi-bit outputs. This is because we can decompose any circuit C with m-bit outputs into m circuits $C_1, \ldots, C_m$ where C_i computes the i^{th} output bit of C. By applying Eval to each circuit $C_1, \ldots, C_m$, the parties can obtain an additive secret sharing of $C_i(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$ for all $i \in [m]$ and concatenate the shares together to obtain an additive secret sharing of $C(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$. Correctness for the multi-bit evaluation case now follows immediately from the correctness definition in Definition 4.1 (e.g., by taking the output length t to be $t = m$; more generally, if we consider evaluating t' circuits with m-bit outputs, then we can take $t = mt'$).

Remark 4.3 (Satisfying a Real-Ideal Simulation Definition)**.** Definition 4.1 considers an indistinguishability-based security notion for reusable multiparty SMS. It was shown in [BJSS25] that this security notion implies the standard real-ideal simulation definition when we additionally compose with a 2-party non-interactive key exchange (NIKE) protocol. At a high-level, the approach is to re-randomize the output shares using a pseudorandom string that is derived using the NIKE protocol. Specifically, as part of the input encoding, the parties generate the first round message for a 2-party NIKE protocol with every other party. Each pair of parties $\mathcal{P}_i$ and $\mathcal{P}_j$ derive a shared PRF key $k_{i,j}$ from the NIKE protocol. Parties $\mathcal{P}_i$ and $\mathcal{P}_j$ use this key to derive a pseudorandom string $r_{i,j}$ by applying the PRF on the input encodings and the circuit C to be evaluated. Each party $\mathcal{P}_i$ now re-randomizes their output share d_i with $r_{i,j}$ for all $j \neq i$ (i.e., their output share in the transformed scheme is $d_i \oplus \bigoplus_{j \neq i} r_{i,j}$). Using a similar proof strategy as in [BJSS25], we can show that if the initial reusable SMS satisfies the indistinguishability-based security notion, then the transformed protocol satisfies the real-ideal simulation definition. We also note that a different transformation given in the work of Mukherjee and Wichs [MW16, §6.2] can also be applied to achieve the real-ideal simulation definition. We observe that we can apply this transformation as the statistical simulatability of the output shares follows directly from the fact that the output shares in the reusable SMS scheme constitute an additive secret sharing of the output.

Reusable two-party SMS for degree-2 polynomials. In our construction of reusable two-party SMS, we will need a reusable two-party SMS that supports the class of degree-2 polynomials over a ring $\mathbb{Z}_p$ (where $p > 2$). Thus, we consider a generalization of Definition 4.1 to $\mathbb{Z}_p$. Specifically, let the input to $\mathcal{P}_0$ be $\mathbf{x}_0 \in \mathbb{Z}_p^N$ and the input to $\mathcal{P}_1$ be $\mathbf{x}_1 \in \mathbb{Z}_p^k$. We define the class of functions $\mathcal{C}_{\mathsf{Q}}$ as follows:

$$\mathcal{C}_{\mathsf{Q}} = \left\{ f \colon \mathbb{Z}_p^N \times \mathbb{Z}_p^k \to \mathbb{Z}_p \mid f(\mathbf{x}_0, \mathbf{x}_1) = \mathbf{a}^\intercal(\mathbf{x}_0 \otimes \mathbf{x}_1) + \mathbf{b}^\intercal \mathbf{x}_0 + \mathbf{c}^\intercal \mathbf{x}_1 + e \right\}, \tag{4.1}$$

where $\mathbf{a} \in \mathbb{Z}_p^{Nk}, \mathbf{b} \in \mathbb{Z}_p^N, \mathbf{c} \in \mathbb{Z}_p^k$, and $e \in \mathbb{Z}_p$. We can now consider a reusable SMS protocol for $\mathcal{C}_{\mathsf{Q}}$ where the Encode algorithm takes inputs over $\mathbb{Z}_p^N$ and $\mathbb{Z}_p^k$ and the Eval algorithm outputs shares of the output over $\mathbb{Z}_p$. Specifically, if $\mathsf{pe}_0 \leftarrow \mathsf{Encode}(\mathsf{crs}, 0, \mathbf{x}_0)$ and $\mathsf{pe}_1 \leftarrow \mathsf{Encode}(\mathsf{crs}, 1, \mathbf{x}_1)$ where $\mathbf{x}_0 \in \mathbb{Z}_p^N$ and $\mathbf{x}_1 \in \mathbb{Z}_p^k$, we require the evaluation algorithms $\mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \mathsf{pe}_1), f, 0, \mathsf{td}_0)$ and $\mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \mathsf{pe}_1), f, 1, \mathsf{td}_1)$ to output shares $d_0, d_1 \in \mathbb{Z}_p$, respectively such that

$$d_0 - d_1 = f(\mathbf{x}_0, \mathbf{x}_1) \in \mathbb{Z}_p. \tag{4.2}$$

We now argue that the reverse trapdoor hash for Boolean circuits from [AMR25, Construction 6.9] implies a reusable SMS for $\mathcal{C}_{\mathsf{Q}}$ with these properties. Consider a Boolean circuit g_i that takes the binary representation of $\mathbf{x}_0$ and $\mathbf{x}_1$ and outputs the i^{th} bit of $\mathbf{x}_0 \otimes \mathbf{x}_1$. The reverse trapdoor hash from [AMR25] outputs a $\mathbb{Z}_2$-secret sharing of $g_i(\mathbf{x}_0, \mathbf{x}_1)$. To do this, [AMR25] first constructs a noisy secret sharing of $g_i(\mathbf{x}_0, \mathbf{x}_1) \cdot q/2$ over $\mathbb{Z}_q$ (where $q = \lambda^{\omega(1)}$) and then applies the $\mathsf{Round}_{q \to 2}$ operation on the noisy shares. This yields additive shares over $\mathbb{Z}_2$ by Lemma 3.1. In our case, we would like a subtractive secret sharing over $\mathbb{Z}_p$ (where $p > 2$). To obtain a subtractive secret sharing over $\mathbb{Z}_p$, we proceed as follows:

- First, we modify [AMR25] to produce a noisy secret sharing of $g_i(\mathbf{x}_0, \mathbf{x}_1) \cdot q/p$ over $\mathbb{Z}_q$. This is done by multiplying the output of LFE.Eval and the matrix $\mathbf{A}$ in [AMR25, Construction 6.9] with $\mathbf{G}^{-1}(q/p \cdot \mathbf{u})$ instead of $\mathbf{G}^{-1}(q/2 \cdot \mathbf{u})$.
- Next, we apply $\mathsf{Round}_{q \to p}$ to the noisy shares to obtain a subtractive secret sharing $d_{0,i}, d_{1,i} \in \mathbb{Z}_p$ of $g_i(\mathbf{x}_0, \mathbf{x}_1)$ over $\mathbb{Z}_p$. As long as $q = p \cdot \lambda^{\omega(1)}$, correctness follows from the local rounding lemma (Lemma 3.1). Namely, $d_{0,i} - d_{1,i} = g_i(\mathbf{x}_0, \mathbf{x}_1) \in \mathbb{Z}_p$.

- Using the subtractive shares over $\mathbb{Z}_p$ of the binary representation of the tensor product $\mathbf{x}_0 \otimes \mathbf{x}_1$, parties $\mathcal{P}_0, \mathcal{P}_1$ can scale them by powers-of-2 to obtain subtractive shares of $\mathbf{x}_0 \otimes \mathbf{x}_1$ over $\mathbb{Z}_p^{Nk}$ (i.e., $\mathbf{d}_0, \mathbf{d}_1 \in \mathbb{Z}_p^{Nk}$ where $\mathbf{d}_0 - \mathbf{d}_1 = \mathbf{x}_0 \otimes \mathbf{x}_1$).

Let $f(\mathbf{x}_0, \mathbf{x}_1) = \mathbf{a}^\intercal(\mathbf{x}_0 \otimes \mathbf{x}_1) + \mathbf{b}^\intercal \mathbf{x}_0 + \mathbf{c}^\intercal \mathbf{x}_1 + e$. Party $\mathcal{P}_0$ can locally compute $\mathbf{a}^\intercal \mathbf{d}_0 + \mathbf{b}^\intercal \mathbf{x}_0 + e$ and party $\mathcal{P}_1$ can locally compute $\mathbf{a}^\intercal \mathbf{d}_1 - \mathbf{c}^\intercal \mathbf{x}_1$. Observe then that

$$(\mathbf{a}^\intercal \mathbf{d}_0 + \mathbf{b}^\intercal \mathbf{x}_0 + e) - (\mathbf{a}^\intercal \mathbf{d}_1 - \mathbf{c}^\intercal \mathbf{x}_1) = \mathbf{a}^\intercal(\mathbf{x}_0 \otimes \mathbf{x}_1) + \mathbf{b}^\intercal \mathbf{x}_0 + \mathbf{c}^\intercal \mathbf{x}_1 + e = f(\mathbf{x}_0, \mathbf{x}_1).$$

The parties now have a subtractive sharing of the output of f that satisfies Eq. (4.2). We summarize this instantiation in the following theorem:

Theorem 4.4 (Reusable Two-Party SMS for Quadratic Functions [AMR25])**.** *Assuming LWE with a sub-exponential modulus-to-noise ratio, there exists an adaptively-secure two-party reusable SMS scheme over $\mathbb{Z}_p$ for the function class $\mathcal{C}_Q$. In particular, the* Eval *function outputs subtractive secret shares over $\mathbb{Z}_p$ (i.e., shares that satisfy Eq. (4.2)).*

Remark 4.5 (Extending to Vector-Valued Functions)**.** We extend the syntax of Theorem 4.4 to vector-valued quadratic functions $f : \mathbb{Z}_p^N \times \mathbb{Z}_p^k \to \mathbb{Z}_p^t$ via component-wise evaluation. Namely, we can write

$$f(\mathbf{x}_0, \mathbf{x}_1) := (f_1(\mathbf{x}_0, \mathbf{x}_1), \ldots, f_t(\mathbf{x}_0, \mathbf{x}_1)).$$

The vector-valued evaluation algorithm simply runs the evaluation algorithm from Theorem 4.4 for each $f_1, \ldots, f_t$ to obtain output shares $d_1, \ldots, d_t \in \mathbb{Z}_p$. The output of the vector-valued evaluation algorithm is then $\mathbf{d} = [d_1, \ldots, d_t]$.

5 Reusable Two-Party SMS Protocol from LWE

In this section, we give our construction of a reusable two-party SMS scheme that supports all bounded-depth Boolean circuits based on the plain LWE assumption (with a sub-exponential modulus-to-noise ratio). Our construction relies on a dual-use technique where we share the same secret key between an algebraic homomorphic MAC and a (leveled) homomorphic encryption scheme. We refer to Section 2.1 for an overview of this construction.

Construction 5.1 (Reusable Two-Party SMS)**.** Let λ be a security parameter and let $C = \{C_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of bounded-depth Boolean circuits indexed by a parameter κ. We assume that each circuit $C \in C_\kappa$ is a Boolean circuit of depth at most $d = d(\kappa)$ and of type $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}$, where $d = d(\kappa)$, $N = N(\kappa)$, and $k = k(\kappa)$ are polynomially-bounded.[10] Our construction relies on the following building blocks:

- A leveled homomorphic encryption scheme $\Pi_{\mathsf{LHE}} = (\mathsf{LHE.KeyGen}, \mathsf{LHE.Enc}, \mathsf{LHE.Eval})$ with nearly-linear decryption, secret key dimension $n = n(\lambda, d(\kappa))$, and modulus $p = p(\lambda, d(\kappa))$. Let $d' = d'(\lambda, \kappa)$ be a bound on the depth of the Boolean circuit computing LHE.Eval. Let $\ell_{\mathsf{ct}} = \ell_{\mathsf{ct}}(\lambda, \kappa)$ be the length of a ciphertext output by LHE.Enc.
- An algebraic homomorphic MAC scheme $\Pi_{\mathsf{AHMAC}} = (\mathsf{AHMAC.Setup}, \mathsf{AHMAC.EvalKey}, \mathsf{AHMAC.EvalTag})$ over $\mathbb{Z}_p^n$.
- A reusable two-party SMS protocol $\Pi_{\mathsf{SMSQ}} = (\mathsf{SMS_Q.Setup}, \mathsf{SMS_Q.Encode}, \mathsf{SMS_Q.Eval})$ for computing the class of degree-2 polynomials $\mathcal{C}_{\mathrm{Q}}$ over $\mathbb{Z}_p$ (see Eq. (4.1) and Theorem 4.4).

We construct our two-party reusable SMS scheme $\Pi_{\mathsf{SMS}} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Eval})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\kappa)$: On input the security parameter λ and the circuit parameter κ, the setup algorithm samples $\mathsf{crs}_{\mathsf{SMSQ}} \leftarrow \mathsf{SMS_Q.Setup}(1^\lambda)$, the local rounding randomness $u \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and outputs $\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u)$.
- $\mathsf{Encode}(\mathsf{crs}, i, \mathbf{x})$: On input the common reference string $\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u)$, an index $i \in \{0,1\}$, and an input $\mathbf{x}$, the encoding algorithm proceeds as follows:

[10] As discussed in Remark 4.2, we can consider circuits with single-bit outputs without loss of generality.

– If $i = 0$, then lift $\mathbf{x} \in \{0,1\}^N$ to $\mathbf{x} \in \mathbb{Z}_p^N$ and compute $(\mathsf{pe}'_0, \mathsf{td}'_0) \leftarrow \mathsf{SMS}_\mathsf{Q}.\mathsf{Encode}(\mathsf{crs}_{\mathsf{SMSQ}}, 0, \mathbf{x})$. Output $\mathsf{pe}_0 = \mathsf{pe}'_0$ and $\mathsf{td}_0 = \mathsf{td}'_0$.

– If $i = 1$, then sample the following:

$$(\mathsf{pk}, \mathbf{s}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, 1^d)$$
$$(\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{AHMAC.Setup}(1^\lambda, 1^{d'}, \mathbf{s})$$
$$(\mathsf{pe}'_1, \mathsf{td}'_1) \leftarrow \mathsf{SMS}_\mathsf{Q}.\mathsf{Encode}(\mathsf{crs}_{\mathsf{SMSQ}}, 1, \mathbf{s}_{\mathsf{in}}).$$

Then, for each $j \in [k]$, compute the following:

* Compute the ciphertext $\mathsf{ct}_j \leftarrow \mathsf{LHE.Enc}(\mathbf{s}, x_j)$. We interpret $\mathsf{ct}_j \in \{0,1\}^{\ell_{\mathsf{ct}}}$ as a binary string.
* Sample an encoding key $\mathbf{k}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell_{\mathsf{ct}}}$ for the ciphertext.
* Compute an algebraic homomorphic MAC $\boldsymbol{\sigma}_j = \mathsf{ct}_j \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}_j$ on ct_j.

Output the public encoding $\mathsf{pe}_1 = \big(\mathsf{pe}'_1, \mathsf{evk}, \mathsf{pk}, \{\mathsf{ct}_j, \boldsymbol{\sigma}_j\}_{j\in[k]}\big)$ and the trapdoor $\mathsf{td}_1 = \big(\mathsf{td}'_1, \mathsf{sk}, \{\mathbf{k}_j\}_{j\in[k]}\big)$.

- $\mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \mathsf{pe}_1), C, b, \mathsf{td})$: On input the common reference string $\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u)$, public encodings $\mathsf{pe}_0 = \mathsf{pe}'_0$ and $\mathsf{pe}_1 = (\mathsf{pe}'_1, \mathsf{evk}, \mathsf{pk}, \{\mathsf{ct}_j, \boldsymbol{\sigma}_j\}_{j\in[k]})$, a circuit $C\colon \{0,1\}^N \times \{0,1\}^k \to \{0,1\}$, a party index $b \in \{0,1\}$, and a decoding trapdoor td, the evaluation algorithm first defines the following functions:

  – Let $g\colon \mathbb{Z}_p^N \times \mathbb{Z}_p^n \to \mathbb{Z}_p^{Nn}$ be the quadratic function where $g(\mathbf{x}_0, \mathbf{s}_{\mathsf{in}}) \coloneqq \mathbf{x}_0 \otimes \mathbf{s}_{\mathsf{in}}$ (we apply Π_{SMSQ} to g component-wise; see Remark 4.5).

  – Let $f_C\colon \{0,1\}^N \times \{0,1\}^{k\ell_{\mathsf{ct}}} \to \{0,1\}^{n \log p}$ be the Boolean circuit

$$f_C(\mathbf{x}_0, (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)) \coloneqq \mathsf{LHE.Eval}(\mathsf{pk}, C(\mathbf{x}_0, \cdot), (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)).$$

  Specifically, f_C runs the evaluation algorithm and outputs the binary representation of the resulting ciphertext.

  – Let $L\colon \mathbb{Z}_p^{n^2 \log p} \to \mathbb{Z}_p$ be the linear function where $L(\mathbf{c} \otimes \mathbf{s}) \coloneqq \mathbf{s}^\intercal \tilde{\mathbf{c}}$, where $\mathbf{c}$ is the binary representation of $\tilde{\mathbf{c}}$.

  Now depending on the value of $b \in \{0,1\}$, the evaluation algorithm proceeds as follows:

  – If $b = 0$, the evaluation algorithm parses $\mathsf{td} = \mathsf{td}'_0$ and computes the tag

$$\tilde{\boldsymbol{\sigma}} = \mathsf{SMS}_\mathsf{Q}.\mathsf{Eval}\big((\mathsf{pe}'_0, \mathsf{pe}'_1), g, 0, \mathsf{td}'_0\big) \in \mathbb{Z}_p^{Nn}.$$

  The algorithm parses $\tilde{\boldsymbol{\sigma}}^\intercal = [\tilde{\boldsymbol{\sigma}}_1^\intercal \mid \cdots \mid \tilde{\boldsymbol{\sigma}}_N^\intercal]$ where $\tilde{\boldsymbol{\sigma}}_j \in \mathbb{Z}_p^n$ for all $j \in [N]$. Then, it computes the tag

$$\boldsymbol{\sigma}_{f_C} = \mathsf{EvalTag}\,(\mathsf{evk}, f_C, (\mathbf{x}_0, (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)), (\tilde{\boldsymbol{\sigma}}_1, \ldots, \tilde{\boldsymbol{\sigma}}_N, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_k)) \in \mathbb{Z}_p^{n^2 \log p}. \tag{5.1}$$

  Output the share $d_0 = \mathsf{Round}_{p\to 2}(L(\boldsymbol{\sigma}_{f_C}) - u) \in \mathbb{Z}_2$.

  – If $b = 1$, the evaluation algorithm parses $\mathsf{td} = \big(\mathsf{td}'_1, \mathsf{sk}, \{\mathbf{k}_j\}_{j\in[k]}\big)$. Then, it computes the encoding key

$$\tilde{\mathbf{k}} = \mathsf{SMS}_\mathsf{Q}.\mathsf{Eval}((\mathsf{pe}'_0, \mathsf{pe}'_1), g, 1, \mathsf{td}'_1).$$

  The algorithm parses $\tilde{\mathbf{k}}^\intercal = [\tilde{\mathbf{k}}_1^\intercal \mid \cdots \mid \tilde{\mathbf{k}}_N^\intercal]$ where $\tilde{\mathbf{k}}_j \in \mathbb{Z}_p^n$ for all $j \in [N]$. Then, it computes the encoding key

$$\mathbf{k}_{f_C} = \mathsf{EvalKey}(\mathsf{sk}, \mathsf{evk}, f_C, (\tilde{\mathbf{k}}_1, \ldots, \tilde{\mathbf{k}}_N, \mathbf{k}_1, \ldots, \mathbf{k}_k)). \tag{5.2}$$

  Finally, it outputs the share $d_1 = \mathsf{Round}_{p\to 2}(-L(\mathbf{k}_{f_C}) + u)$.

Theorem 5.2 (Correctness). *If Π_{LHE}, Π_{AHMAC}, and Π_{SMSQ} are correct and $p > \lambda^{\omega(1)}$, then Construction 5.1 is correct.*

Proof. Take any size bound $s = s(\lambda) < 2^\lambda$ and any polynomial $t = t(\lambda)$ denoting the number of circuits to be evaluated (cf. Definition 4.1). Take any security parameter $\lambda \in \mathbb{N}$, circuit family parameter $\kappa \in \mathbb{N}$, inputs $\mathbf{x}_0 \in \{0,1\}^N$ and $\mathbf{x}_1 \in \{0,1\}^k$ and a collection of circuits $C_1, \ldots, C_t \in \mathcal{C}_\kappa$ where $|C_i| \le s(\lambda)$. Let $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$. Write $\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u)$ where $\mathsf{crs}_{\mathsf{SMSQ}} \leftarrow \mathsf{SMS}_{\mathsf{Q}}.\mathsf{Setup}(1^\lambda)$ and $u \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. Let

$$(\mathsf{pe}_0, \mathsf{td}_0) \leftarrow \mathsf{Encode}(\mathsf{crs}, 0, \mathbf{x}_0)$$
$$(\mathsf{pe}_1, \mathsf{td}_1) \leftarrow \mathsf{Encode}(\mathsf{crs}, 1, \mathbf{x}_1).$$

Fix a function $C \in \{C_1, \ldots, C_t\}$ and compute $d_0 := \mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \mathsf{pe}_1), C, 0, \mathsf{td}_0)$ and $d_1 := \mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \mathsf{pe}_1), C, 1, \mathsf{td}_1)$. Consider the value of $d_0 \oplus d_1$:

- First, we can write
$$\mathsf{pe}_0 = \mathsf{pe}'_0 \qquad \mathsf{pe}_1 = \left(\mathsf{pe}'_1, \mathsf{evk}, \mathsf{pk}, \{\mathsf{ct}_j, \boldsymbol{\sigma}_j\}_{j\in[k]}\right)$$
$$\mathsf{td}_0 = \mathsf{td}'_0 \qquad \mathsf{td}_1 = \left(\mathsf{td}'_1, \mathsf{sk}, \{\mathbf{k}_j\}_{j\in[k]}\right).$$

- Consider the values of $\tilde{\boldsymbol{\sigma}}$ and $\tilde{\mathbf{k}}$ from the computations of d_0 and d_1. By construction,
$$\tilde{\boldsymbol{\sigma}} = \mathsf{SMS}_{\mathsf{Q}}.\mathsf{Eval}\left((\mathsf{pe}'_0, \mathsf{pe}'_1), g, 0, \mathsf{td}'_0\right) \in \mathbb{Z}_p^{Nn}$$
$$\tilde{\mathbf{k}} = \mathsf{SMS}_{\mathsf{Q}}.\mathsf{Eval}((\mathsf{pe}'_0, \mathsf{pe}'_1), g, 1, \mathsf{td}'_1) \in \mathbb{Z}_p^{Nn}.$$
Since $\mathsf{pe}'_0, \mathsf{pe}'_1$ are encodings of $\mathbf{x}_0 \in \mathbb{Z}_p^N$ and $\mathbf{s}_{\mathsf{in}} \in \mathbb{Z}_p^n$, respectively, and Π_{SMSQ} outputs subtractive shares (see Eq. (4.2)), we conclude by correctness of Π_{SMSQ} that, with overwhelming probability,
$$\tilde{\boldsymbol{\sigma}} - \tilde{\mathbf{k}} = g(\mathbf{x}_0, \mathbf{s}_{\mathsf{in}}) = \mathbf{x}_0 \otimes \mathbf{s}_{\mathsf{in}}.$$
Expanding component-wise, for all $i \in [N]$, we have $\tilde{\boldsymbol{\sigma}}_i = x_{0,i} \cdot \mathbf{s}_{\mathsf{in}} + \tilde{\mathbf{k}}_i$.

- Next, consider the value $\boldsymbol{\sigma}_{f_C}$ and $\mathbf{k}_{f_C}$ from the computation of d_0 and d_1 (see Eqs. (5.1) and (5.2)). Since $\boldsymbol{\sigma}_j = \mathsf{ct}_j \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}_j$, by correctness of Π_{AHMAC}, with overwhelming probability,
$$\boldsymbol{\sigma}_{f_C} = f_C(\mathbf{x}_0, (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)) \otimes \mathbf{s} + \mathbf{k}_{f_C} = \mathbf{c} \otimes \mathbf{s} + \mathbf{k}_{f_C},$$
where $\mathbf{c}$ is the binary representation of $\tilde{\mathbf{c}} = \mathsf{LHE}.\mathsf{Eval}(\mathsf{pk}, C(\mathbf{x}_0, \cdot), (\mathsf{ct}_1, \ldots, \mathsf{ct}_k))$.

- Next, L is a linear function, so
$$L(\boldsymbol{\sigma}_{f_C}) - L(\mathbf{k}_{f_C}) = L(\mathbf{c} \otimes \mathbf{s}) = \mathbf{s}^\intercal \tilde{\mathbf{c}}.$$
By correctness of Π_{LHE}, with overwhelming probability, $\mathbf{s}^\intercal \tilde{\mathbf{c}} = C(\mathbf{x}_0, \mathbf{x}_1) \cdot \lfloor p/2 \rfloor + e$ where $|e| < \sqrt{p}$.

- Since $u \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, with overwhelming probability over the choice of u, it follows from Lemma 3.1 that $d_0 \oplus d_1 = C(\mathbf{x}_0, \mathbf{x}_1)$ except with probability $O(1/\sqrt{p}) = \mathsf{negl}(\lambda)$.

Thus, with overwhelming probability over the randomness of Setup and Encode, we conclude that for any $C \in \{C_1, \ldots, C_t\}$ correctness holds with overwhelming probability. Since $t = \mathsf{poly}(\lambda)$, the claim now holds by a union bound. □

Theorem 5.3 (Security). *If Π_{LHE} is CPA-secure, Π_{AHMAC} is selectively secure, and Π_{SMSQ} is selectively secure, then Construction 5.1 is adaptively secure.*

Proof. By construction, adaptive security for $\mathcal{P}_0$ follows directly from adaptive security of Π_{SMSQ}. It suffices to show adaptive security for $\mathcal{P}_1$. We proceed via a sequence of hybrids. Let $\mathcal{A}$ be an adversary for the security game.

- Hyb_0: This is the security experiment where $b = 0$. Specifically, the adversary declares the circuit-policy parameter 1^κ. The challenger then samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ and gives crs to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs two inputs $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^k$. The challenger then samples $(\mathsf{pe}_1, \mathsf{td}_1) \leftarrow \mathsf{Encode}(\mathsf{crs}, 1, \mathbf{x}_0)$ and replies with pe_1.

Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment. Specifically, in this experiment, the challenger samples

$$\begin{aligned} \mathsf{crs}_{\mathsf{SMSQ}} &\leftarrow \mathsf{SMS}_{\mathsf{Q}}.\mathsf{Setup}(1^\lambda) \\ u &\xleftarrow{\mathrm{R}} \mathbb{Z}_p \\ (\mathsf{pk}, \mathbf{s}) &\leftarrow \mathsf{LHE.KeyGen}(1^\lambda, 1^d) \\ (\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) &\leftarrow \mathsf{AHMAC.Setup}(1^\lambda, 1^{d'}, \mathbf{s}) \\ (\mathsf{pe}'_1, \mathsf{td}'_1) &\leftarrow \mathsf{SMS}_{\mathsf{Q}}.\mathsf{Encode}(\mathsf{crs}_{\mathsf{SMSQ}}, 1, \mathbf{s}_{\mathsf{in}}). \end{aligned}$$

Then, for each $j \in [k]$, it computes the following:

- Compute the ciphertext $\mathsf{ct}_j \leftarrow \mathsf{LHE.Enc}(\mathbf{s}, x_{0,j})$. We interpret $\mathsf{ct}_j \in \{0,1\}^{\ell_{\mathsf{ct}}}$ as a binary string.
- Sample an encoding key $\mathbf{k}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell_{\mathsf{ct}}}$ for the ciphertext.
- Compute an algebraic homomorphic MAC $\boldsymbol{\sigma}_j = \mathsf{ct}_j \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}_j$ on ct_j.

The challenger sets the common reference string and the public encoding to be

$$\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u) \quad \text{and} \quad \mathsf{pe}_1 = \left(\mathsf{pe}'_1, \mathsf{evk}, \mathsf{pk}, \{\mathsf{ct}_j, \boldsymbol{\sigma}_j\}_{j\in[k]}\right). \tag{5.3}$$

- Hyb_1: Same as Hyb_0, except during Setup, the challenger now samples $(\mathsf{pe}'_1, \mathsf{td}'_1) \leftarrow \mathsf{SMS}_{\mathsf{Q}}.\mathsf{Encode}(\mathsf{crs}_{\mathsf{SMSQ}}, 1, \mathbf{0}^n)$.
- Hyb_2: Same as Hyb_1, except the challenger uses the simulator for Π_{AHMAC} to generate evk and the tags $\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_k$. Specifically, let $(\mathcal{S}_1, \mathcal{S}_2)$ be the simulator for Π_{AHMAC}. Then the challenger computes $(\mathsf{evk}, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^{d'})$ and, for each $j \in [k]$, samples $(\boldsymbol{\sigma}_j, \mathsf{st}) \leftarrow \mathcal{S}_2(\mathsf{st}, 1^{\ell_{\mathsf{ct}}})$.
- Hyb_3: Same as Hyb_2, except the challenger generates $\mathsf{ct}_j \leftarrow \mathsf{LHE.Enc}(\mathbf{s}, x_{1,j})$ for all $j \in [k]$.
- Hyb_4: Same as Hyb_3, except the challenger generates the algebraic homomorphic MAC as in the real scheme. Specifically, the challenger in this experiment samples $(\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{AHMAC.Setup}(1^\lambda, 1^{d'}, \mathbf{s})$ and $\mathbf{k}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell_{\mathsf{ct}}}$ for all $j \in [k]$, and sets $\boldsymbol{\sigma}_j = \mathsf{ct}_j \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}_j$.
- Hyb_5: Same as Hyb_4, except the challenger constructs $\mathsf{crs}_{\mathsf{SMSQ}}$ as in the real scheme. Specifically, the challenger now samples $(\mathsf{pe}'_1, \mathsf{td}'_1) \leftarrow \mathsf{SMS}_{\mathsf{Q}}.\mathsf{Encode}(\mathsf{crs}_{\mathsf{SMSQ}}, 1, \mathbf{s}_{\mathsf{in}})$. This is the security experiment where $b = 1$.

We write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output distribution of Hyb_i with adversary $\mathcal{A}$. We now argue that each adjacent pair of hybrids is computationally indistinguishable:

- Hyb_0 and Hyb_1 are computationally indistinguishable by (selective) security of Π_{SMSQ}. We show this in Lemma 5.4.
- Hyb_1 and Hyb_2 are computationally indistinguishable by security of Π_{AHMAC}. We show this in Lemma 5.5.
- Hyb_2 and Hyb_3 are computationally indistinguishable by CPA-security of Π_{LHE}. We show this in Lemma 5.6.
- Hyb_3 and Hyb_4 are computationally indistinguishable by security of Π_{AHMAC}. This claim follows by an analogous argument as the proof of Lemma 5.5.
- Hyb_4 and Hyb_5 are computationally indistinguishable by (selective) security of Π_{SMSQ}. This claim follows by an analogous argument as the proof of Lemma 5.4.

Lemma 5.4. *If Π_{SMSQ} is selectively secure, then $\mathsf{Hyb}_0(\mathcal{A})$ and $\mathsf{Hyb}_1(\mathcal{A})$ are computationally indistinguishable.*

Proof. Suppose $|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| = \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an efficient *selective* adversary $\mathcal{B}$ for Π_{SMSQ} (with party index 1):

- Algorithm $\mathcal{B}$ starts by sampling $(\mathsf{pk}, \mathbf{s}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, 1^d)$ and $(\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{AHMAC.Setup}(1^\lambda, 1^{d'}, \mathbf{s})$. It provides the challenge messages $(\mathbf{s}_{\mathsf{in}}, \mathbf{0}^n)$ to the Π_{SMSQ} challenger and receives $\mathsf{crs}_{\mathsf{SMSQ}}$ and pe'_1.
- Algorithm $\mathcal{B}$ samples $u \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and gives $\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u)$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs a pair of messages $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^k$.
- For each $j \in [k]$, algorithm $\mathcal{B}$ computes $\mathsf{ct}_j \leftarrow \mathsf{LHE.Enc}(\mathbf{s}, x_{0,j})$, $\mathbf{k}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell_{\mathsf{ct}}}$, and $\boldsymbol{\sigma}_j = \mathsf{ct}_j \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}_j$.
- Algorithm $\mathcal{B}$ gives $\mathsf{pe}_1 = \left(\mathsf{pe}'_1, \mathsf{evk}, \mathsf{pk}, \{\mathsf{ct}_j, \boldsymbol{\sigma}_j\}_{j\in[k]}\right)$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

By construction, if pe'_1 is an encoding of $\mathbf{s}_{\mathsf{in}}$, then algorithm $\mathcal{B}$ perfectly simulates Hyb_0 whereas if pe'_1 is an encoding of $\mathbf{0}^n$, then algorithm $\mathcal{B}$ perfectly simulates Hyb_1. Thus, algorithm $\mathcal{B}$ breaks security of Π_{SMSQ} with advantage ε. □

Lemma 5.5. *If Π_{AHMAC} is selectively secure, then $\mathsf{Hyb}_1(\mathcal{A})$ and $\mathsf{Hyb}_2(\mathcal{A})$ are computationally indistinguishable.*

Proof. Suppose $|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]| = \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ for the algebraic homomorphic MAC:

- Algorithm $\mathcal{B}$ starts by sampling $\mathsf{crs}_{\mathsf{SMSQ}} \leftarrow \mathsf{SMS_Q.Setup}(1^\lambda)$ and $(\mathsf{pe}'_1, \mathsf{td}'_1) \leftarrow \mathsf{SMS_Q.Encode}(\mathsf{crs}_{\mathsf{SMSQ}}, 1, \mathbf{0}^n)$.
- Algorithm $\mathcal{B}$ samples $u \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and gives $\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u)$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs a pair of messages $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^k$.
- Algorithm $\mathcal{B}$ now samples $(\mathsf{pk}, \mathbf{s}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, 1^d)$. Then, for each $j \in [k]$, it constructs $\mathsf{ct}_j \leftarrow \mathsf{LHE.Enc}(\mathbf{s}, x_{0,j})$.
- Algorithm $\mathcal{B}$ then provides the depth bound $1^{d'}$, the global offset $\mathbf{s}$, and the queries $\mathsf{ct}_1, \ldots, \mathsf{ct}_k \in \{0,1\}^{\ell_{\mathsf{ct}}}$ to the challenger. The challenger responds with the evaluation key evk and tags $\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_k$.
- Algorithm $\mathcal{B}$ gives $\mathsf{pe}_1 = (\mathsf{pe}'_1, \mathsf{evk}, \mathsf{pk}, \{\mathsf{ct}_j, \boldsymbol{\sigma}_j\}_{j\in[k]})$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

By construction, if the challenger samples $(\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{AHMAC.Setup}(1^\lambda, 1^{d'}, \mathbf{s})$ and $\boldsymbol{\sigma}_j = \mathsf{ct}_j \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}_j$ where $\mathbf{k}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell_{\mathsf{ct}}}$, then algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_1. Alternatively, if it samples $(\mathsf{evk}, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^{d'})$ and $(\boldsymbol{\sigma}_j, \mathsf{st}) \leftarrow \mathcal{S}_2(\mathsf{st}, 1^{\ell_{\mathsf{ct}}})$, then it perfectly simulates an execution of Hyb_2. Thus, algorithm $\mathcal{B}$ breaks security of Π_{AHMAC} with the same advantage ε. □

Lemma 5.6. *If Π_{LHE} is CPA-secure, then $\mathsf{Hyb}_2(\mathcal{A})$ and $\mathsf{Hyb}_3(\mathcal{A})$ are computationally indistinguishable.*

Proof. Suppose $|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]| = \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ for the leveled homomorphic encryption scheme (with depth bound d):

- Algorithm $\mathcal{B}$ starts by sampling $\mathsf{crs}_{\mathsf{SMSQ}} \leftarrow \mathsf{SMS_Q.Setup}(1^\lambda)$ and $(\mathsf{pe}'_1, \mathsf{td}'_1) \leftarrow \mathsf{SMS_Q.Encode}(\mathsf{crs}_{\mathsf{SMSQ}}, 1, \mathbf{0}^n)$.
- Algorithm $\mathcal{B}$ samples $u \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and gives $\mathsf{crs} = (1^d, \mathsf{crs}_{\mathsf{SMSQ}}, u)$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs a pair of messages $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^k$.
- Algorithm $\mathcal{B}$ now queries the CPA-security challenger on the pair of messages $(x_{0,1}, x_{1,1}), \ldots, (x_{0,k}, x_{1,k})$. It receives the public key pk and ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ (encrypting the bits of either $\mathbf{x}_0$ or $\mathbf{x}_1$).
- Algorithm $\mathcal{B}$ then runs $(\mathsf{evk}, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^{d'})$ and $(\boldsymbol{\sigma}_j, \mathsf{st}) \leftarrow \mathcal{S}_2(\mathsf{st}, 1^{\ell_{\mathsf{ct}}})$ for all $j \in [k]$.
- Algorithm $\mathcal{B}$ gives $\mathsf{pe}_1 = (\mathsf{pe}'_1, \mathsf{evk}, \mathsf{pk}, \{\mathsf{ct}_j, \boldsymbol{\sigma}_j\}_{j\in[k]})$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

By construction, if $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ are encryptions of the bits of $\mathbf{x}_0$, then algorithm $\mathcal{B}$ perfectly simulates Hyb_2. If they are encryptions of the bits of $\mathbf{x}_1$, then algorithm $\mathcal{B}$ perfectly simulates Hyb_3. Thus, algorithm $\mathcal{B}$ breaks security of Π_{LHE} with the same advantage ε. □

Theorem 5.3 now follows by a hybrid argument. □

Theorem 5.7 (Succinctness). *If Π_{AHMAC} and Π_{SMSQ} satisfy succinctness, then Construction 5.1 is also succinct.*

Proof. We have $|\mathsf{crs}_{\mathsf{SMSQ}}| = \mathsf{poly}(\lambda, |\mathbf{s}_{\mathsf{in}}|)$ where $\mathbf{s}_{\mathsf{in}} \in \mathbb{Z}_p^n$. Since $n \log p = \mathsf{poly}(\lambda, d)$, this means $|\mathsf{crs}| = \mathsf{poly}(\lambda, d)$. Next, succinctness of Π_{SMSQ} implies that $|\mathsf{pe}'_0| = \mathsf{poly}(\lambda, d)$ and $|\mathsf{pe}'_1| = \mathsf{poly}(\lambda, d)$. By definition, $|\mathsf{pk}| = \mathsf{poly}(\lambda, d)$ and succinctness of the algebraic MAC implies that $|\mathsf{evk}| = \mathsf{poly}(\lambda, d') = \mathsf{poly}(\lambda, d)$ since $d' = \mathsf{poly}(\lambda, d)$. Each ciphertext ct_i and each tag $\boldsymbol{\sigma}_i$ has size $\mathsf{poly}(\lambda, d)$. Thus, putting everything together, the size of pe_1 is $\mathsf{poly}(\lambda, d, k)$. □

Remark 5.8 (Sub-Exponential Security). We note that if each of the underlying building blocks ($\Pi_{\mathsf{LHE}}, \Pi_{\mathsf{AHMAC}}, \Pi_{\mathsf{SMSQ}}$) in Construction 5.1 is sub-exponentially-secure, then the proof of Theorem 5.3 shows that Construction 5.1 also satisfies sub-exponential security.

6 Reusable Multiparty SMS Protocol from Obfuscation

In this section, we show how to bootstrap a reusable two-party SMS protocol to the multiparty setting using indistinguishability obfuscation.

6.1 Building Blocks

We start by reviewing the main building blocks we will use in our reusable multiparty SMS construction.

Definition 6.1 (Indistinguishability Obfuscation [BGI+01, GGH+13]). An indistinguishability obfuscation scheme for a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is an efficient algorithm $i\mathcal{O}(\cdot, \cdot)$ with the following two properties:

- **Correctness.** For all $\lambda \in \mathbb{N}$, and all circuits $C \in \mathcal{C}_\lambda$ where $C \colon \{0,1\}^n \to \{0,1\}$ and all inputs $x \in \{0,1\}^n$, we have that
$$\Pr\left[C(x) = \tilde{C}(x) : \tilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)\right] = 1.$$
- **Security.** For a bit $b \in \{0,1\}$ and a security parameter λ, we define the program indistinguishability game between an adversary $\mathcal{A}$ and a challenger as follows:
  - On input the security parameter 1^λ, the adversary outputs two Boolean circuits $C_0, C_1 \in \mathcal{C}_\lambda$.
  - If there exists an input x such that $C_0(x) \neq C_1(x)$, then the experiment outputs 0 and halts. Otherwise, the challenger replies with $i\mathcal{O}(1^\lambda, C_b)$.
  - The adversary outputs a bit $b' \in \{0,1\}$ which is the output of the experiment.

  We say $i\mathcal{O}$ is secure if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,
$$|\Pr[b' = 1 : b = 0] - \Pr[b' = 1 : b = 1]| = \mathsf{negl}(\lambda).$$

Definition 6.2 (Puncturable PRF [BW13, KPTZ13, BGI14]). A puncturable family of pseudorandom functions with input length $n = n(\lambda)$ and output length $m = m(\lambda)$ is a tuple of efficient algorithms $\Pi_{\mathsf{PRF}} = (\mathsf{KeyGen}, \mathsf{Punc}, \mathsf{Eval})$ with the following syntax:

- $\mathsf{KeyGen}(1^\lambda) \to k$: On input the security parameter λ, the key-generation algorithm outputs a PRF key k
- $\mathsf{Eval}(k, x) \to y$: On input a key k and an input $x \in \{0,1\}^n$, the function returns a value $y \in \{0,1\}^m$.
- $\mathsf{Punc}(k, x) \to k_x$: On input the key k and an input $x \in \{0,1\}^n$, the puncturing algorithm outputs a punctured key k_x.

We require Π_{PRF} satisfy the following properties:

- **Punctured correctness:** For all $x^* \in \{0,1\}^n$, all $\lambda \in \mathbb{N}$, all keys k in the support of $\mathsf{KeyGen}(1^\lambda)$, and all keys k_{x^*} in the support of $\mathsf{Punc}(k, x^*)$, we have that

$$\forall x \neq x^* : \mathsf{Eval}(k, x) = \mathsf{Eval}(k_{x^*}, x).$$

- **Pseudorandomness at the punctured point.** For a security parameter λ, an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$, we define the (selective) pseudorandomness game as follows

  - On input the security parameter 1^λ, the adversary outputs a point $x^* \in \{0,1\}^n$ and sends x^* to the challenger.
  - The challenger samples the PRF key $k \leftarrow \mathsf{KeyGen}(1^\lambda)$ and computes the punctured key $k_{x^*} \leftarrow \mathsf{Punc}(k, x^*)$. If $b = 0$, the challenger computes $y = \mathsf{Eval}(k, x^*)$; if $b = 1$, the challenger samples $y \xleftarrow{R} \{0,1\}^m$. The challenger then sends (k_{x^*}, y) to the adversary.
  - The adversary outputs bit $b' \in \{0,1\}$, which is the output of the experiment.

  We say that the puncturable PRF satisfies (selective) pseudorandomness if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda).$$

Definition 6.3 (Public-Key Encryption)**.** Let λ be a security parameter. A public-key encryption scheme with message space $\mathcal{M} = \{0,1\}^*$ consists of a tuple of efficient algorithms $\Pi_{\mathsf{PKE}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input the security parameter λ, the key-generation algorithm outputs a public key pk and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, \mu) \to \mathsf{ct}$: On input the public key pk and a message $\mu \in \{0,1\}^*$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mu$: On input the secret key sk and a ciphertext ct, the decryption algorithm outputs a message μ.

We require Π_{PKE} to satisfy the following properties:

- **Correctness:** For all $\lambda \in \mathbb{N}$ and all messages $\mu \in \{0,1\}^*$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mu : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu)\end{array}\right] = 1.$$

- **Semantic security:** For a security parameter λ and a bit $b \in \{0,1\}$, we define the semantic security game for an adversary $\mathcal{A}$ as follows:

  - The challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives $(1^\lambda, \mathsf{pk})$ to the adversary $\mathcal{A}$.
  - The adversary $\mathcal{A}$ outputs a pair of messages (μ_0, μ_1) with $|\mu_0| = |\mu_1|$. The challenger replies with $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_b)$.
  - The adversary outputs a bit $b' \in \{0,1\}$ which is the output of the experiment.

  The scheme is semantically secure if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda).$$

Definition 6.4 (Statistical Simulation-Sound NIZK [GGH+13])**.** A simulation-sound NIZK proof for NP is a triple of efficient algorithms $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}_{\mathsf{NIZK}}$: On input the security parameter λ, the setup algorithm outputs a common reference string $\mathsf{crs}_{\mathsf{NIZK}}$.

- $\mathsf{Prove}(\mathsf{crs}_{\mathsf{NIZK}}, \mathcal{R}, x, w) \to \pi$: On input the CRS $\mathsf{crs}_{\mathsf{NIZK}}$, an NP relation $\mathcal{R}\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ represented as a Boolean circuit, a statement $x \in \{0,1\}^n$, and a witness $w \in \{0,1\}^h$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}_{\mathsf{NIZK}}, \mathcal{R}, x, \pi) \to b$: On input the CRS $\mathsf{crs}_{\mathsf{NIZK}}$, an NP relation $\mathcal{R}\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ represented as a Boolean circuit, a statement $x \in \{0,1\}^n$, and a proof π, the verifier outputs a bit $b \in \{0,1\}$.

We require Π_{NIZK} to satisfy the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$, all NP relations $\mathcal{R}\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, all statements $x \in \{0,1\}^n$, and all witnesses $w \in \{0,1\}^h$ where $\mathcal{R}(x,w) = 1$, we have that

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}_{\mathsf{NIZK}}, \mathcal{R}, x, \pi) = 1 : \begin{array}{c} \mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}_{\mathsf{NIZK}}, \mathcal{R}, x, w) \end{array}\right] = 1.$$

- **Non-adaptive computational zero-knowledge:** For a security parameter λ, an adversary $\mathcal{A}$, a simulator $\mathcal{S}$, and a bit $b \in \{0,1\}$, we define the non-adaptive computational zero-knowledge experiment as follows:
  - On input the security parameter 1^λ, the adversary outputs an NP relation $\mathcal{R}\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a witness $w \in \{0,1\}^h$.
  - The challenger first checks that $\mathcal{R}(x,w) = 1$ and responds with $\perp$ if not. Then, the challenger proceeds as follows:
    - If $b = 0$, the challenger samples $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{Setup}(1^\lambda)$ and $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}_{\mathsf{NIZK}}, \mathcal{R}, x, w)$.
    - If $b = 1$, the challenger samples $(\mathsf{crs}_{\mathsf{NIZK}}, \pi) \leftarrow \mathcal{S}(1^\lambda, \mathcal{R}, x)$.

    The challenger gives $(\mathsf{crs}_{\mathsf{NIZK}}, \pi)$ to $\mathcal{A}$.
  - At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is also the output of the experiment.

  We say that Π_{NIZK} satisfies non-adaptive computational zero-knowledge if there exists an efficient simulator $\mathcal{S}$ such that for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda)$$

  in the non-adaptive computational zero-knowledge experiment defined above.

- **Statistical simulation soundness:** Let $\mathcal{S}$ be the zero-knowledge simulator from the non-adaptive computational zero-knowledge definition. For a security parameter λ and an adversary $\mathcal{A}$ we define the statistical simulation soundness game as follows:
  - On input the security parameter 1^λ, the adversary starts by outputting an NP relation $\mathcal{R}\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ and a statement $x \in \{0,1\}^n$.
  - The challenger samples and gives $(\mathsf{crs}_{\mathsf{NIZK}}, \pi) \leftarrow \mathcal{S}(1^\lambda, \mathcal{R}, x)$ to $\mathcal{A}$.
  - Algorithm $\mathcal{A}$ outputs a statement $x' \in \{0,1\}^n$ and a proof π'.
  - The output is $b = 1$ if $x' \neq x$ and $\mathsf{Verify}(\mathsf{crs}_{\mathsf{NIZK}}, \mathcal{R}, x', \pi') = 1$, and there does not exist any $w' \in \{0,1\}^h$ such that $\mathcal{R}(x', w') = 1$. Otherwise, the output is $b = 0$.

  We say that Π_{NIZK} satisfies statistical simulation soundness if for all (possibly unbounded) adversaries $\mathcal{A}$, there exists a negligible function such that $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the statistical simulation soundness security experiment.

The work of [GGH+13] shows how to construct statistical simulation-sound NIZK proofs by combining a vanilla NIZK proof system and a perfectly binding commitment scheme.

6.2 Constructing Reusable Multiparty SMS from Obfuscation

We now show how to bootstrap a reusable two-party SMS protocol to the multiparty setting using an indistinguishability obfuscation scheme (in combination with a puncturable PRF and a simulation-sound NIZK proof).

Construction 6.5 (Reusable Multiparty SMS)**.** Let λ denote the security parameter and $C = \{C_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of bounded-depth Boolean circuits where each circuit $f \in C_\kappa$ is a Boolean circuit of depth at most $d = d(\kappa)$ and of type $f\colon \{0,1\}^N \times (\{0,1\}^\ell)^n \to \{0,1\}$, where $d = d(\kappa)$, $N = N(\kappa)$, $\ell = \ell(\kappa)$, and $n = n(\kappa)$ are polynomially-bounded.[11] We define two additional security parameters $\lambda_1 = \mathsf{poly}(\lambda, n, \ell)$ and $\lambda_2 = \mathsf{poly}(\lambda, d, \log N, n, \ell)$ which we will set in the security analysis. Our reusable multiparty SMS construction relies on the following building blocks:

- Let $\Pi_{\mathsf{rSMS}} = (\mathsf{rSMS.Setup}, \mathsf{rSMS.Encode}, \mathsf{rSMS.Eval})$ be a reusable two-party SMS scheme for C with sub-exponential security.
- Let $\kappa'(\lambda, \kappa)$ be an injective and efficiently-computable mapping, and let $C' = \{C'_{\kappa'}\}_{\kappa'\in\mathbb{N}}$ be a family of Boolean circuits that includes circuits of the form

$$G(f, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1})) := \mathsf{rSMS.Eval}\left(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}), f, 1, \mathsf{td}_{\mathsf{rSMS},1}\right),$$

  where $f \in C_\kappa$ and $\mathsf{crs}_{\mathsf{rSMS}}$ is a CRS in the support of $\mathsf{rSMS.Setup}(1^\lambda, 1^\kappa)$.
- Let $\Pi_{\mathsf{SMS}} = (\mathsf{SMS.Setup}, \mathsf{SMS.Encode}, \mathsf{SMS.Decode})$ be a two-party SMS scheme for the circuit class C' with sub-exponential security.
- Let $i\mathcal{O}$ be an indistinguishability obfuscation scheme for Boolean circuits that supports the Boolean circuits appearing in the description of the construction as well as the programs appearing in the security proof. We require that $i\mathcal{O}$ satisfies sub-exponential security.
- Let $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a semantically-secure public-key encryption scheme.
- Let $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ be a statistical simulation-sound NIZK proof (Definition 6.4) for the NP relation $\mathcal{R}_{\mathsf{eq}}$ defined as follows:

$$\mathcal{R}_{\mathsf{eq}}\big((\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1), (m, r_0, r_1)\big) = 1 \iff \forall b \in \{0,1\} : \mathsf{ct}_b = \mathsf{PKE.Enc}(\mathsf{pk}_b, m; r_b).$$

- Let $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Punc}, \mathsf{PRF.Eval})$ be a puncturable PRF family (Definition 6.2). We require Π_{PRF} to satisfy sub-exponential security.

For ease of exposition, we assume without loss of generality that the PKE.Enc, rSMS.Encode, and SMS.Encode algorithms above take λ bits of randomness (when instantiated with security parameter λ).[12] We construct a reusable multiparty SMS $\Pi_{\mathsf{rSMS}} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Eval})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\kappa)$: On input the security parameter λ and the circuit-family parameter κ, the setup algorithm proceeds as follows:
  1. Sample $\mathsf{crs}_{\mathsf{rSMS}} \leftarrow \mathsf{rSMS.Setup}(1^{\lambda_1}, 1^\kappa)$ and $\mathsf{crs}_{\mathsf{SMS}} \leftarrow \mathsf{SMS.Setup}(1^{\lambda_1}, 1^{\kappa'})$ where $\kappa' = \kappa'(\lambda_1, \kappa)$.
  2. For each $i \in [n]$,
     (a) Sample $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ for each $b \in \{0,1\}$.
     (b) Sample $\mathsf{crs}_{\mathsf{NIZK},i} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
  3. Sample $K \leftarrow \mathsf{PRF.KeyGen}(1^{\lambda_1})$.

[11] In this construction, we use f to denote the Boolean circuit in order to distinguish it from the (obfuscated) circuit $\widetilde{C}$ in the CRS. Additionally, we refer to Remark 4.2 on how to support circuits with multiple output bits.

[12] This is without loss of generality since we can always take the randomness to an algorithm to be the seed of a pseudorandom generator (PRG) and use the PRG to expand the seed into however many bits of randomness the algorithm requires. By a standard hybrid argument, this preserves the security of the underlying scheme.

4. Compute $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, C)$ where C is the following circuit:

Inputs: $(f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$.
Hardwired: $K, \{\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{sk}_{i,0}, \mathsf{crs}_{\mathsf{NIZK},i}\}_{i\in[n]}, \mathsf{crs}_{\mathsf{rSMS}}, \mathsf{crs}_{\mathsf{SMS}}$.

(a) For every $i \in [n]$, parse pe_i as $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}, \pi_i)$.

(b) For every $i \in [n]$,

i. Check $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK},i}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}), \pi_i) = 1$.

ii. If the check fails, output $\perp$.

(c) For every $i \in [n]$,

i. Compute $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,0}, \mathsf{ct}_{i,0})$.

ii. Compute $\mathsf{mask}_i = \mathsf{PRF.Eval}(k_i, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.

(d) Compute $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.

(e) Compute $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) = \mathsf{rSMS.Encode}\big(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}}\big)$.

(f) Compute $(\mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1}) = \mathsf{SMS.Encode}\big(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}}\big)$.

(g) Compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \mathsf{td}_{\mathsf{SMS},1})$.

(h) Output $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.

Figure 1: Description of C.

5. Sample randomness $r \xleftarrow{\mathrm{R}} \{0,1\}^{\lambda_1}$.
6. Output the common reference string

$$\mathsf{crs} = (\mathsf{crs}_{\mathsf{rSMS}}, \mathsf{crs}_{\mathsf{SMS}}, \{\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{crs}_{\mathsf{NIZK},i}\}_{i\in[n]}, \widetilde{C}, r). \tag{6.1}$$

- $\mathsf{Encode}(\mathsf{crs}, i, x_i)$: On input the common reference string crs (with components parsed according to Eq. (6.1)), the party index $i \in [0, n]$, and the input x_i, the encoding algorithm proceeds as follows:
  - If $i = 0$, then compute $(\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{td}_{\mathsf{rSMS},0}) \leftarrow \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 0, x_0)$ and output the public encoding $\mathsf{pe}_0 = \mathsf{pe}_{\mathsf{rSMS},0}$ and the trapdoor $\mathsf{td}_0 = (\mathsf{td}_{\mathsf{rSMS},0}, r)$.
  - If $i \in [n]$, then proceed as follows:
    - Sample $k_i \leftarrow \mathsf{PRF.KeyGen}(1^{\lambda_2})$ and set $m_i = (x_i, k_i)$.
    - Compute $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, m_i; r_{i,b})$ for each $b \in \{0,1\}$ where $r_{i,b} \xleftarrow{\mathrm{R}} \{0,1\}^{\lambda}$.
    - Compute $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK},i}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}), (m_i, r_{i,0}, r_{i,1}))$.
    - Output the public encoding $\mathsf{pe}_i = (\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}, \pi_i)$ and the trapdoor $\mathsf{td}_i = k_i$.
- $\mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n), f, i, \mathsf{td}_i)$: On input the common reference string crs (with components parsed according to Eq. (6.1)), public encodings $\mathsf{pe}_0, \ldots, \mathsf{pe}_n$, the function $f\colon \{0,1\}^N \times (\{0,1\}^\ell)^n \to \{0,1\}$, an index $i \in [0, n]$, and a trapdoor td_i, the evaluation algorithm proceeds as follows:
  1. Compute $(f_{\mathsf{SMS}}, \mathsf{td}_{\mathsf{SMS},0}) = \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 0, (f, G); r)$ where G is the circuit

$$G(f, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1})) := \mathsf{rSMS.Eval}\left(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}), f, 1, \mathsf{td}_{\mathsf{rSMS},1}\right).$$

  2. If $i = 0$, parse $\mathsf{pe}_0 = \mathsf{pe}_{\mathsf{rSMS},0}$ and the trapdoor $\mathsf{td}_0 = (\mathsf{td}_{\mathsf{rSMS},0}, r)$ and then proceed as follows:

(a) Compute $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, d_1') \leftarrow \widetilde{C}(f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$.
(b) Compute $d_{\mathsf{SMS},0} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 0, \mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},0})$.
(c) Compute $d_{\mathsf{rSMS},0} = \mathsf{rSMS.Eval}(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}), f, 0, \mathsf{td}_{\mathsf{rSMS},0})$.
(d) Output $d_0 = d_{\mathsf{rSMS},0} \oplus d_{\mathsf{SMS},0} \oplus d_1'$.

3. If $i \neq 0$, parse $\mathsf{td}_i = k_i$ and output $d_i = \mathsf{PRF.Eval}(k_i, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.

Theorem 6.6 (Correctness). *If $i\mathcal{O}$, Π_{PKE}, Π_{SMS}, Π_{rSMS} are correct, Π_{NIZK} is complete, and Π_{PRF} satisfies pseudorandomness, then Construction 6.5 is correct.*

Proof. Take any security parameter λ and circuit-family parameter κ, inputs $x_0 \in \{0,1\}^N$, $x_1, \ldots, x_n \in \{0,1\}^\ell$. Consider any sequence of functions $f_1, \ldots, f_t \in C_\kappa$ for $t = \mathsf{poly}(\lambda)$. Take f to be an arbitrary function in the sequence $f_1, \ldots, f_t$. Then, we have the following:

- Let $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$, $(\mathsf{pe}_i, \mathsf{td}_i) \leftarrow \mathsf{Encode}(\mathsf{crs}, i, x_i)$, and $d_i = \mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n), f, i, \mathsf{td}_i)$ for all $i \in [0, n]$.
- Consider the value of $d_0 \oplus \cdots \oplus d_n$. By construction, we can write

$$\mathsf{crs} = (\mathsf{crs}_{\mathsf{rSMS}}, \mathsf{crs}_{\mathsf{SMS}}, \{\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{crs}_{\mathsf{NIZK},i}\}_{i\in[n]}, \widetilde{C}, r).$$

  First, consider the value of $d_0 = \mathsf{Eval}(\mathsf{crs}, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n), f, 0, \mathsf{td}_0)$. By correctness of $i\mathcal{O}$, this means

$$(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, d_1') = C(f_{\mathsf{SMS}}, \mathsf{pe}_0, \mathsf{pe}_1, \ldots, \mathsf{pe}_n).$$

  By completeness of Π_{NIZK}, this means

$$\begin{aligned}(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) &= \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}})\\ (\mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1}) &= \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}}),\end{aligned}$$

  where $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$. The evaluation algorithm then computes

$$\begin{aligned}d_{\mathsf{SMS},0} &= \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 0, \mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},0})\\ d_{\mathsf{rSMS},0} &= \mathsf{rSMS.Eval}(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}), f, 0, \mathsf{td}_{\mathsf{rSMS},0}).\end{aligned}$$

  Finally, the evaluation algorithm sets

$$\begin{aligned}d_0 &= d_{\mathsf{rSMS},0} \oplus d_{\mathsf{SMS},0} \oplus d_1'\\ \forall i \in [n] : d_i &= \mathsf{PRF.Eval}(k_i, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)).\end{aligned}$$

Our analysis now proceeds in two steps:

- First, we show that if $r_{\mathsf{rSMS}}, r_{\mathsf{SMS}} \xleftarrow{R} \{0,1\}^{\lambda_1}$, then with overwhelming probability over the choice of $r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}$, $\mathsf{crs}_{\mathsf{rSMS}}$, and $\mathsf{crs}_{\mathsf{SMS}}$, we have that $d_0 \oplus d_1 \oplus \cdots \oplus d_n = f(x_0, x_1, \ldots, x_n)$.
- In the real scheme, the values are derived by computing $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$ where $K \leftarrow \mathsf{PRF.KeyGen}(1^{\lambda_1})$. If correctness holds with overwhelming probability for the values of $d_0, \ldots, d_n$ computed via the above procedure when $r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}$ are uniform random, then correctness will continue to hold with overwhelming probability when $r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}$ are derived from the PRF. This is because the circuit C, the Encode algorithm, and the Decode algorithm are all efficiently-computable and moreover, the values of $d_0, \ldots, d_n$ and $f(x_0, x_1, \ldots, x_n)$ can all be computed given only f, the inputs $x_0, \ldots, x_n$, r_{rSMS}, and r_{SMS}, and in particular, *without* knowledge of the PRF key K.

It suffices to show that correctness holds with overwhelming probability when $r_{\mathsf{rSMS}}, r_{\mathsf{SMS}} \xleftarrow{R} \{0,1\}^{\lambda_1}$. By definition of C,

$$d_1' = d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n,$$

where $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \mathsf{td}_{\mathsf{SMS},1})$ and $\mathsf{mask}_i = \mathsf{PRF.Eval}(k_i, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$, where the latter holds by completeness of Π_{NIZK} and correctness of Π_{PKE}. By correctness of Π_{SMS}, with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{SMS}}$ and the randomness $r, r_{\mathsf{SMS}} \xleftarrow{R} \{0,1\}^{\lambda_1}$, we have

$$\begin{aligned} d_{\mathsf{SMS},0} \oplus d_{\mathsf{SMS},1} &= G(f, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1})) \\ &= \mathsf{rSMS.Eval}(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}), f, 1, \mathsf{td}_{\mathsf{rSMS},1}) \\ &= \mathsf{rSMS.Eval}(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}), f, 1, \mathsf{td}_{\mathsf{rSMS},1}), \end{aligned}$$

using the fact that $\mathsf{pe}_0 = \mathsf{pe}_{\mathsf{rSMS},0}$. Next, we have

$$\begin{aligned} (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{td}_{\mathsf{rSMS},0}) &\leftarrow \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 0, x_0) \\ (\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) &= \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}}). \end{aligned}$$

With overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{rSMS}}$ and the randomness $r_{\mathsf{rSMS}} \xleftarrow{R} \{0,1\}^{\lambda_1}$,

$$d_{\mathsf{rSMS},0} \oplus \mathsf{rSMS.Eval}(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}), f, 1, \mathsf{td}_{\mathsf{rSMS},1}) = f(x_0, x_1, \ldots, x_n),$$

where $d_{\mathsf{rSMS},0} = \mathsf{rSMS.Eval}(\mathsf{crs}_{\mathsf{rSMS}}, (\mathsf{pe}_{\mathsf{rSMS},0}, \mathsf{pe}_{\mathsf{rSMS},1}), f, 0, \mathsf{td}_{\mathsf{rSMS},0})$. Combining the above relations, we have

$$\begin{aligned} d_0 = d_{\mathsf{rSMS},0} \oplus d_{\mathsf{SMS},0} \oplus d_1' &= d_{\mathsf{rSMS},0} \oplus d_{\mathsf{SMS},0} \oplus d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n \\ &= f(x_0, x_1, \ldots, x_n) \oplus d_1 \oplus \cdots \oplus d_n. \end{aligned}$$

Correspondingly, this means that with overwhelming probability, $d_0 \oplus d_1 \oplus \cdots \oplus d_n = f(x_0, \ldots, x_n)$, as required. By a union bound over the functions $f_1, \ldots, f_t$, correctness holds except with negligible probability. □

Theorem 6.7 (Security)**.** *Suppose Π_{PKE} is semantically secure, Π_{NIZK} satisfies non-adaptive computational zero-knowledge and statistical simulation soundness, that Π_{rSMS}, Π_{SMS}, $i\mathcal{O}$, are sub-exponentially secure, and that Π_{PRF} satisfies punctured correctness and pseudorandomness at the punctured point with sub-exponential security. Then Construction 6.5 is selectively secure.*

Proof. Security for $\mathcal{P}_0$ follows directly from the security of the two-party succinct rSMS scheme. It suffices to consider security for $\mathcal{P}_i$ for some $i \in [n]$. We prove this via a hybrid argument. We start with the real distribution where the input encoding pe_i for $\mathcal{P}_i$ is computed according to the real scheme (via $\mathsf{Encode}(\mathsf{crs}, i, x_i^*)$ on an adversary-chosen input x_i^*) and show that this is computationally indistinguishable from an encoding pe_i that is computed using a procedure which is *independent* of the input x_i^*. This is sufficient to prove security.[13]

- Hyb_0: This is the selective security experiment where the encoding pe_i is computed using the real encoding algorithm:
  - On input the security parameter, the adversary starts by declaring the circuit-family parameter 1^κ, the party index $i \in [n]$, and the input $x_i^* \in \{0,1\}^\ell$.
  - The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ as follows:
    - Sample $\mathsf{crs}_{\mathsf{rSMS}} \leftarrow \mathsf{rSMS.Setup}(1^{\lambda_1}, 1^\kappa)$ and $\mathsf{crs}_{\mathsf{SMS}} \leftarrow \mathsf{SMS.Setup}(1^{\lambda_1}, 1^{\kappa'})$ where $\kappa' = \kappa'(\lambda_1, \kappa)$.
    - For each $j \in [n]$, sample $(\mathsf{pk}_{j,b}, \mathsf{sk}_{j,b}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ for each $b \in \{0,1\}$ and $\mathsf{crs}_{\mathsf{NIZK},j} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.

[13] Technically, this is a slightly different security definition than that in Definition 4.1 which asks that $\mathsf{Encode}(\mathsf{crs}, i, x_{i,0})$ be computationally indistinguishable from $\mathsf{Encode}(\mathsf{crs}, i, x_{i,1})$, where $x_{i,0}, x_{i,1}$ are inputs of the adversary's choosing. The definition we consider here is equivalent since we show that the distribution of $\mathsf{Encode}(\mathsf{crs}, i, x_i^*)$ can be simulated without knowledge of x_i^*.

* Sample $K \leftarrow \mathsf{PRF.KeyGen}(1^{\lambda_1})$ and compute $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, C)$, where C is the circuit defined in Fig. 1.
* Sample randomness $r \xleftarrow{R} \{0,1\}^{\lambda_1}$.

The challenger sets $\mathsf{crs} = (\mathsf{crs}_{\mathsf{rSMS}}, \mathsf{crs}_{\mathsf{SMS}}, \{\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{crs}_{\mathsf{NIZK},j}\}_{j\in[n]}, \widetilde{C}, r)$.

– Next, the challenger computes the public encoding pe_i by running $\mathsf{Encode}(\mathsf{crs}, i, x_i^*)$. Specifically, the challenger proceeds as follows:

* Sample $k_i^* \leftarrow \mathsf{PRF.KeyGen}(1^{\lambda_2})$ and set $m_i = (x_i^*, k_i^*)$.
* For each $b \in \{0,1\}$, sample $r_{i,b} \xleftarrow{R} \{0,1\}^{\lambda}$ and compute $\mathsf{ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, m_i; r_{i,b})$.
* Compute $\pi_i \leftarrow \mathsf{NIZK.Prove}\big(\mathsf{crs}_{\mathsf{NIZK},i}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}), (m_i, r_{i,0}, r_{i,1})\big)$.

The challenger replies to $\mathcal{A}$ with the common reference string crs and the public encoding $\mathsf{pe}_i = (\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}, \pi_i)$.

– Algorithm $\mathcal{A}$ outputs a bit $b \in \{0,1\}$ which is the output of the experiment.

- Hyb_1: Same as Hyb_0, except the challenger replaces $\mathsf{crs}_{\mathsf{NIZK},i}$ and π_i with simulated values. Formally, let $\mathcal{S}$ be the zero-knowledge simulator associated with Π_{NIZK} (Definition 6.4). The challenger now samples

$$(\mathsf{crs}_{\mathsf{NIZK},i}, \pi_i) \leftarrow \mathcal{S}(1^{\lambda}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{ct}_{i,0}, \mathsf{ct}_{i,1})).$$

  Note that in this experiment, the challenger's behavior no longer requires knowledge of the encryption randomness $r_{i,0}, r_{i,1}$.

  The output distributions of Hyb_0 and Hyb_1 are computationally indistinguishable by non-adaptive computational zero-knowledge of Π_{NIZK}.

- Hyb_2: Same as Hyb_1, except the challenger now samples $\mathsf{ct}_{i,1} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,1}, \bot)$.

  By design, the challenger's behavior in Hyb_1 does not depend on the secret key $\mathsf{sk}_{i,1}$ or the encryption randomness $r_{i,1}$. Thus, the output distributions of Hyb_1 and Hyb_2 are computationally indistinguishable by semantic security of Π_{PKE}.

- Hyb_3: Same as Hyb_2, except the challenger sets $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1}) := (\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1})$ and replaces the obfuscated circuit by $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, C_1)$ where C_1 is the circuit shown in Fig. 2:

Inputs: $(f_{\mathsf{sms}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$.
Hardwired: K, $\{\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{crs}_{\mathsf{nizk},j}\}_{j\in[n]}$, $\{\mathsf{sk}_{j,0}\}_{j\neq i}$, $\mathsf{sk}_{i,1}$, $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, (x_i^*, k_i^*), $\mathsf{crs}_{\mathsf{rsms}}$, $\mathsf{crs}_{\mathsf{sms}}$.

1. For every $j \in [n]$, parse pe_j as $(\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}, \pi_j)$.
2. For every $j \in [n]$,
   (a) Check if $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{nizk},j}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}), \pi_j) = 1$.
   (b) If the check fails, output $\bot$.
3. For every $j \in [n]$,
   (a) If $j = i$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, set $(x_i, k_i) = (x_i^*, k_i^*)$.
   (b) Else, if $j = i$ but $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) \neq (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, compute $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$.
   (c) Else, compute $(x_j, k_j) = \mathsf{PKE.Dec}(\mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})$.
   (d) Compute $\mathsf{mask}_j = \mathsf{PRF.Eval}(k_j, (f_{\mathsf{sms}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
4. Compute $(r_{\mathsf{rsms}}, r_{\mathsf{sms}}) = \mathsf{PRF.Eval}(K, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
5. Compute $(\mathsf{pe}_{\mathsf{rsms},1}, \mathsf{td}_{\mathsf{rsms},1}) = \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rsms}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rsms}})$.
6. Compute $(\mathsf{pe}_{\mathsf{sms},1}, \mathsf{td}_{\mathsf{sms},1}) = \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{sms}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rsms},1}, \mathsf{td}_{\mathsf{rsms},1}); r_{\mathsf{sms}})$.
7. Compute $d_{\mathsf{sms},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{sms}}, 1, f_{\mathsf{sms}}, \mathsf{td}_{\mathsf{sms},1})$.
8. Output $(\mathsf{pe}_{\mathsf{rsms},1}, \mathsf{pe}_{\mathsf{sms},1}, d_{\mathsf{sms},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.

Figure 2: Description of C_1.

In Lemma 6.8, we show that by security of $i\mathcal{O}$ and statistical simulation soundness of Π_{NIZK}, the output distributions of Hyb_2 and Hyb_3 are computationally indistinguishable. Note that in this experiment, the challenger's behavior no longer requires knowledge of the secret key $\mathsf{sk}_{i,0}$ (since the challenger decrypts using $\mathsf{sk}_{i,1}$ instead).

- Hyb_4: Same as Hyb_3, except the challenger now samples $\mathsf{ct}_{i,0} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,0}, \bot)$. At this point, the public encoding pe_i no longer depends on the input x_i^*. However, the program C_1 has x_i^* hardcoded.

  By design, the challenger's behavior in Hyb_3 does not depend on the secret key $\mathsf{sk}_{i,0}$ or the encryption randomness $r_{i,0}$. Thus, the output distributions of Hyb_3 and Hyb_4 are computationally indistinguishable by semantic security of Π_{PKE}.

- Hyb_{4,j^*}: Same as Hyb_4, except the challenger changes the distribution of the obfuscated program $\widetilde{C}$. Specifically, let $\prec$ be a lexicographic ordering on the set of inputs $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$ for which $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, and let inp_{j^*} denote the $(j^*)^{\text{th}}$ such input. In this experiment, the challenger now computes $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, C'_{j^*})$, where C'_{j^*} is the circuit shown in Fig. 3:

Inputs: $(f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$.
Hardwired: K, $\{\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{crs}_{\mathsf{NIZK},j}\}_{j\in[n]}$, $\{\mathsf{sk}_{j,0}\}_{j\neq i}$, $\mathsf{sk}_{i,1}$, $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, (x_i^*, k_i^*), inp_{j^*}, $\mathsf{crs}_{\mathsf{rSMS}}$, $\mathsf{crs}_{\mathsf{SMS}}$.

1. For every $j \in [n]$, parse pe_j as $(\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}, \pi_j)$.
2. For every $j \in [n]$,
   (a) Check if $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK},j}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}), \pi_j) = 1$.
   (b) If the check fails, output $\perp$.
3. For every $j \in [n]$,
   (a) If $j = i$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, set $(x_i, k_i) = (x_i^*, k_i^*)$.
   (b) Else, if $j = i$, but $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) \neq (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, compute $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$.
   (c) Else, compute $(x_j, k_j) = \mathsf{PKE.Dec}(\mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})$.
   (d) Compute $\mathsf{mask}_j = \mathsf{PRF.Eval}(k_j, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
4. Compute $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
5. If $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) \prec \mathsf{inp}_{j^*}$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, then reset $x_i = \perp$.
6. Compute $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) = \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}})$.
7. Compute $(\mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1}) = \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}})$.
8. Compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \mathsf{td}_{\mathsf{SMS},1})$.
9. Output $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.

Figure 3: Description of C'_{j^*}.

By construction, the programs C'_1 and C_1 are functionally equivalent and hence, it follows from $i\mathcal{O}$ security that the output distribution in Hyb_4 is computationally indistinguishable from that in $\mathsf{Hyb}_{4,1}$.

In Lemma 6.9, we show that for all $j^* \geq 1$, Hyb_{4,j^*} is computationally indistinguishable from Hyb_{4,j^*+1}.

- Hyb_5: Same as Hyb_4, except the challenger again changes the distribution of the obfuscated circuit $\widetilde{C}$. Specifically, let M be the total number of inputs $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$ for which $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, and let $C_2 = C'_{M+1}$ be the circuit that sets $x_i = \perp$ on every input $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$ with $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$. In this experiment, the challenger then replaces the obfuscated circuit with $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, C_2)$. By design, crs and pe_i in this experiment no longer depend on the value of x_i^*.

  In C'_{M+1}, every input $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$ with $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$ satisfies $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) \prec \mathsf{inp}_{M+1}$, so the conditional reset in Fig. 3 always sets $x_i = \perp$; hence the hardwired x_i^* is never used. Therefore C'_{M+1} and C_2 are functionally equivalent, and $\mathsf{Hyb}_{4,M+1}$ and Hyb_5 are computationally indistinguishable by $i\mathcal{O}$ security.

  Since the distribution of $(\mathsf{crs}, \mathsf{pe}_i)$ in this experiment is independent of the input x_i^*, the adversary's view in this experiment no longer depends on the private input x_i^*, as required.

As usual, we write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output of an execution of Hyb_i with adversary $\mathcal{A}$. Below, we give formal proofs for indistinguishability of adjacent pairs of hybrid experiments (for the transitions that were not already sketched above).

Lemma 6.8. *Suppose Π_{PKE} is correct, $i\mathcal{O}$ is secure, and Π_{NIZK} satisfies statistical simulation soundness (Definition 6.4). Then $\mathsf{Hyb}_2(\mathcal{A})$ and $\mathsf{Hyb}_3(\mathcal{A})$ are computationally indistinguishable.*

Proof. The only difference between these two experiments is the distribution of $\widetilde{C}$. In Hyb_2, $\widetilde{C}$ is an obfuscation of the circuit C, whereas in Hyb_3, it is an obfuscation of the circuit C_1. We first show that these two circuits are functionally equivalent (with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{NIZK},i}$). Take an input $(f_{\mathsf{sms}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$ and parse $\mathsf{pe}_j = (\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}, \pi_j)$ for each $j \in [n]$. Suppose there exists an index $j \in [n]$ where

$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK},j}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}), \pi_j) = 0.$$

In this case, both C and C_1 output $\bot$ and the circuits agree. Suppose now that all of the proofs verify. By construction, the two circuits compute $(x_j, k_j) = \mathsf{PKE.Dec}(\mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})$ identically for all $j \neq i$. Hence the only possible source of disagreement is in how (x_i, k_i) is computed. We consider two cases:

- Suppose $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$. In Hyb_2, the challenger sampled $\mathsf{ct}_{i,0} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{i,0}, (x_i^*, k_i^*); r_{i,0})$, and in Hyb_3 it sets $\mathsf{ct}'_{i,0} := \mathsf{ct}_{i,0}$. By correctness of Π_{PKE}, $\mathsf{PKE.Dec}(\mathsf{sk}_{i,0}, \mathsf{ct}'_{i,0}) = (x_i^*, k_i^*)$. Thus, circuit C computes $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,0}, \mathsf{ct}_{i,0}) = (x_i^*, k_i^*)$, while circuit C_1 directly sets $(x_i, k_i) = (x_i^*, k_i^*)$. The two circuits agree on this input.
- Suppose $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) \neq (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$. In this case, the circuit C computes $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,0}, \mathsf{ct}_{i,0})$, while the circuit C_1 computes $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$. We argue that these are equal with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{NIZK},i}$. In Hyb_2 and Hyb_3, the challenger generates $(\mathsf{crs}_{\mathsf{NIZK},i}, \pi_i)$ using the NIZK simulator $\mathcal{S}$ on the statement

  $$(\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1}).$$

  By statistical simulation soundness of Π_{NIZK} (Definition 6.4), with overwhelming probability over $\mathsf{crs}_{\mathsf{NIZK},i}$, there does not exist any accepting proof of a false statement other than $(\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$. Since $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) \neq (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$ and π_i is accepting, it follows that there exists (m, r_0, r_1) such that $\mathsf{ct}_{i,b} = \mathsf{PKE.Enc}(\mathsf{pk}_{i,b}, m; r_b)$ for both $b \in \{0, 1\}$. By correctness of Π_{PKE}, this means $\mathsf{PKE.Dec}(\mathsf{sk}_{i,0}, \mathsf{ct}_{i,0}) = m = \mathsf{PKE.Dec}(\mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$, and the two circuits agree on this input.

We conclude that C and C_1 are functionally equivalent with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{NIZK},i}$. Indistinguishability of Hyb_2 and Hyb_3 now follows from $i\mathcal{O}$ security. □

Lemma 6.9. *Suppose $i\mathcal{O}$, Π_{SMS}, and Π_{rSMS} are sub-exponentially secure, and Π_{PRF} is correct and satisfies pseudorandomness at the punctured point with sub-exponential security. Then, for all $j^* \in [M]$, $\mathsf{Hyb}_{4,j^*}(\mathcal{A})$ and $\mathsf{Hyb}_{4,j^*+1}(\mathcal{A})$ are computationally indistinguishable, where M is the total number of inputs $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$ for which $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$.*

Proof. We define a new sequence of hybrid experiments:

- Hyb'_1 : Same as Hyb_{4,j^*}, except the challenger precomputes the output of the obfuscated circuit on the input inp_{j^*} and hardwires the precomputed values into a modified circuit. Specifically, the challenger proceeds as follows:
  1. Parse $\mathsf{inp}_{j^*} = (\mathsf{pe}_0, \ldots, \mathsf{pe}_n)$ and parse $\mathsf{pe}_j = (\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}, \pi_j)$ for each $j \in [n]$.
  2. For every $j \in [n]$, check that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK},j}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}), \pi_j) = 1$. If any check fails, set $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}}) = \bot$.
  3. If all checks pass, then compute the following:
     (a) Set $(x_i, k_i) = (x_i^*, k_i^*)$ and for each $j \neq i$, let $(x_j, k_j) = \mathsf{PKE.Dec}(\mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})$.
     (b) Compute $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K, \mathsf{inp}_{j^*})$.
     (c) Compute $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) = \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}})$.
     (d) Compute $(\mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1}) = \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}})$.

     Finally, the challenger sets $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}}) = (\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1})$.

The challenger then samples $\widetilde{C} \leftarrow iO(1^{\lambda_2}, C'_{j^*,1})$, where $C'_{j^*,1}$ is the circuit shown in Fig. 4 with the precomputed tuple $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$ hardwired:

Inputs: $(f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$.
Hardwired: K, $\{\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{crs}_{\mathsf{NIZK},j}\}_{j\in[n]}$, $\{\mathsf{sk}_{j,0}\}_{j\neq i}$, $\mathsf{sk}_{i,1}$, $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, (x_i^*, k_i^*), inp_{j^*}, $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$, $\mathsf{crs}_{\mathsf{rSMS}}$, $\mathsf{crs}_{\mathsf{SMS}}$.

1. For every $j \in [n]$, parse pe_j as $(\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}, \pi_j)$.
2. For every $j \in [n]$,
   (a) Check if $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK},j}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}), \pi_j) = 1$.
   (b) If the check fails, output $\perp$.
3. Else, for every $j \in [n]$,
   (a) If $j = i$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, set $(x_i, k_i) = (x_i^*, k_i^*)$.
   (b) Else, if $j = i$, but $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) \neq (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, compute $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$.
   (c) Else, compute $(x_j, k_j) = \mathsf{PKE.Dec}(\mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})$.
   (d) Compute $\mathsf{mask}_j = \mathsf{PRF.Eval}(k_j, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
4. If $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) = \mathsf{inp}_{j^*}$, then do the following:
   (a) If $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}}) = \perp$, output $\perp$.
   (b) Otherwise, compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$.
   (c) Output $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.
5. Else, compute $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K, (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
6. If $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) \prec \mathsf{inp}_{j^*}$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, reset $x_i = \perp$.
7. Compute $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) = \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}})$.
8. Compute $(\mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1}) = \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}})$.
9. Compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \mathsf{td}_{\mathsf{SMS},1})$.
10. Output $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.

Figure 4: Description of $C'_{j^*,1}$.

By construction, C'_{j^*} and $C'_{j^*,1}$ are functionally equivalent (specifically, $C'_{j^*,1}$ simply hardcodes the value that would be computed by C'_{j^*} on input inp_{j^*}). By iO security, $\mathsf{Hyb}_{4,j^*}(\mathcal{A})$ and $\mathsf{Hyb}'_1(\mathcal{A})$ are computationally indistinguishable.

- Hyb'_2: Same as Hyb'_1, except the challenger samples a punctured key $K[\mathsf{inp}_{j^*}] \leftarrow \mathsf{PRF.Punc}(K, \mathsf{inp}_{j^*})$ and hardwires this punctured key in place of the unpunctured key K in the obfuscated circuit.

The modified circuit is functionally equivalent to $C'_{j^*,1}$ since $C'_{j^*,1}$ never evaluates the PRF using K at the punctured point inp_{j^*} (its output on input inp_{j^*} is taken from the precomputed tuple $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$). On all other points, the evaluation using the punctured key coincides with the evaluation using the real key

(by correctness of Π_{PRF}). Since the two programs are functionally equivalent, we can appeal to $i\mathcal{O}$ security to conclude that the output distributions $\mathsf{Hyb}'_1(\mathcal{A})$ and $\mathsf{Hyb}'_2(\mathcal{A})$ are computationally indistinguishable.

- $\mathsf{Hyb}'_{3,t}$: Same as Hyb'_2, except the challenger once again changes the distribution of the obfuscated circuit $\widetilde{C}$. Specifically, let $\prec'$ be a lexicographic ordering on the set of possible f_{SMS} values. Let $f_{\mathsf{SMS}}[t]$ denote the t^{th} input in this ordering. In this experiment, the challenger hardwires $f_{\mathsf{SMS}}[t]$ into the obfuscated circuit. Specifically, the challenger now samples $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, C'_{j^*,2,t})$, where $C'_{j^*,2,t}$ is the circuit shown in Fig. 5 below:

Inputs: $(f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$.
Hardwired: $K[\mathsf{inp}_{j^*}]$, $\{\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{crs}_{\mathsf{NIZK},j}\}_{j\in[n]}$, $\{\mathsf{sk}_{j,0}\}_{j\neq i}$, $\mathsf{sk}_{i,1}$, $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, (x^*_i, k^*_i), inp_{j^*}, $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$, $f_{\mathsf{SMS}}[t]$, $\mathsf{crs}_{\mathsf{rSMS}}$, $\mathsf{crs}_{\mathsf{SMS}}$.

1. For every $j \in [n]$, parse pe_j as $(\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}, \pi_j)$.
2. For every $j \in [n]$,
   (a) Check if $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK},j}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}), \pi_j) = 1$.
   (b) If the check fails, output $\perp$.
3. Else, for every $j \in [n]$,
   (a) If $j = i$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, set $(x_i, k_i) = (x^*_i, k^*_i)$.
   (b) Else, if $j = i$, but $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) \neq (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, compute $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$.
   (c) Else, compute $(x_j, k_j) = \mathsf{PKE.Dec}(\mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})$.
   (d) Compute $\mathsf{mask}_j = \mathsf{PRF.Eval}(k_j, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
4. If $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) = \mathsf{inp}_{j^*}$, then do the following:
   (a) If $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}}) = \perp$, output $\perp$.
   (b) Else if $f_{\mathsf{SMS}} \prec' f_{\mathsf{SMS}}[t]$, set $d_{\mathsf{SMS},1} = 0$.
   (c) Otherwise, compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$.
   (d) Output $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.
5. Else, compute $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K[\mathsf{inp}_{j^*}], (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
6. If $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) \prec \mathsf{inp}_{j^*}$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, reset $x_i = \perp$.
7. Compute $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) = \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}})$.
8. Compute $(\mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1}) = \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}})$.
9. Compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \mathsf{td}_{\mathsf{SMS},1})$.
10. Output $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.

Figure 5: Description of $C'_{j^*,2,t}$.

By construction, the functionality of $C'_{j^*,2,1}$ coincides with that of the circuit obfuscated in Hyb'_2 since the new condition $f_{\mathsf{SMS}} \prec' f_{\mathsf{SMS}}[1]$ is never triggered (namely, $f_{\mathsf{SMS}}[1]$ is the lexicographic-first entry in the total order of values of f_{SMS}). Thus, by $i\mathcal{O}$ security, hybrids $\mathsf{Hyb}'_2(\mathcal{A})$ and $\mathsf{Hyb}'_{3,1}(\mathcal{A})$ are computationally indistinguishable.

In Lemma 6.10, we show that $\mathsf{Hyb}'_{3,t}(\mathcal{A})$ and $\mathsf{Hyb}'_{3,t+1}(\mathcal{A})$ are computationally indistinguishable for all

$t \in [2^{|f_{\mathsf{SMS}}|}]$.

- Hyb'_4 : Same as $\mathsf{Hyb}'_{3,M'}$, except the challenger sets the hardwired trapdoor $\widetilde{\mathsf{td}_{\mathsf{SMS},1}} = \bot$ in the obfuscated circuit. Here, we set $M' = 2^{|f_{\mathsf{SMS}}|} + 1$ and let $f_{\mathsf{SMS}}[M']$ denote a virtual sentinel value that succeeds every actual f_{SMS} value in the ordering $\prec'$ (i.e., $f_{\mathsf{SMS}} \prec' f_{\mathsf{SMS}}[M']$ for every f_{SMS}).

  In $\mathsf{Hyb}'_{3,M'}$, the trapdoor $\widetilde{\mathsf{td}_{\mathsf{SMS},1}}$ is no longer needed to generate $d_{\mathsf{SMS},1}$. Specifically, since $f_{\mathsf{SMS}} \prec' f_{\mathsf{SMS}}[M']$ for every f_{SMS}, the program $C'_{j^*,2,M'}$ always sets $d_{\mathsf{SMS},1} = 0$ on input inp_{j^*}. Thus, $\mathsf{Hyb}'_{3,M'}(\mathcal{A})$ and $\mathsf{Hyb}'_4(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security.

- Hyb'_5 : Same as Hyb'_4, except the challenger samples $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) \xleftarrow{R} \{0,1\}^{\lambda_1} \times \{0,1\}^{\lambda_1}$ uniformly at random when computing the precomputed values $\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}$.

  Hybrids $\mathsf{Hyb}'_4(\mathcal{A})$ and $\mathsf{Hyb}'_5(\mathcal{A})$ are computationally indistinguishable since Π_{PRF} satisfies pseudorandomness at the punctured point. Note that the entirety of Hyb'_4 and Hyb'_5 can be simulated just given the evaluations $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}})$ and the punctured key $K[\mathsf{inp}_{j^*}]$.

- Hyb'_6 : Same as Hyb'_5, except the challenger samples $(\widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}}) \leftarrow \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\bot, \bot, \bot))$ when computing the precomputed values.

  The trapdoor $\widetilde{\mathsf{td}_{\mathsf{SMS},1}}$ is not used elsewhere in this hybrid (it is set to $\bot$ in the hardwired tuple by Hyb'_4). Thus, $\mathsf{Hyb}'_5(\mathcal{A})$ and $\mathsf{Hyb}'_6(\mathcal{A})$ are computationally indistinguishable by security of Π_{SMS}.

- Hyb'_7 : Same as Hyb'_6, except the challenger samples

$$(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \mathsf{td}_{\mathsf{rSMS},1}) \leftarrow \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_{i-1}, \bot, x_{i+1}, \ldots, x_n))$$

  when computing the precomputed values.

  The trapdoor $\mathsf{td}_{\mathsf{rSMS},1}$ is not used in generating the remaining components of this hybrid. Thus, $\mathsf{Hyb}'_6(\mathcal{A})$ and $\mathsf{Hyb}'_7(\mathcal{A})$ are computationally indistinguishable by security of Π_{rSMS}.

- Hyb'_8 : Same as Hyb'_7, except the challenger reverts the change in Hyb'_6 and samples

$$(\widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}}) \leftarrow \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}})$$

  when computing the precomputed values.

  As in the $\mathsf{Hyb}'_5 \to \mathsf{Hyb}'_6$ transition, the trapdoor $\widetilde{\mathsf{td}_{\mathsf{SMS},1}}$ is not used elsewhere in this hybrid. Thus, $\mathsf{Hyb}'_7(\mathcal{A})$ and $\mathsf{Hyb}'_8(\mathcal{A})$ are computationally indistinguishable by security of Π_{SMS}.

- Hyb'_9 : Same as Hyb'_8, except the challenger reverts the change in Hyb'_5 and computes

$$(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K, \mathsf{inp}_{j^*})$$

  when computing the precomputed values (rather than sampling uniformly).

  As in the $\mathsf{Hyb}'_4 \to \mathsf{Hyb}'_5$ transition, indistinguishability of $\mathsf{Hyb}'_8(\mathcal{A})$ and $\mathsf{Hyb}'_9(\mathcal{A})$ follows from the pseudorandomness at the punctured point property of Π_{PRF}.

- Hyb'_{10} : Same as Hyb'_9, except the challenger reverts the change in Hyb'_4 and restores $\widetilde{\mathsf{td}_{\mathsf{SMS},1}}$ to its precomputed value in the hardwired tuple of the obfuscated circuit (rather than $\bot$).

  As in the $\mathsf{Hyb}'_{3,2^{|f_{\mathsf{SMS}}|}} \to \mathsf{Hyb}'_4$ transition, the trapdoor is not used by the obfuscated circuit, so $\mathsf{Hyb}'_9(\mathcal{A})$ and $\mathsf{Hyb}'_{10}(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security.

- Hyb'_{11} : Same as Hyb'_{10}, except the challenger reverts the chain of changes in $\mathsf{Hyb}'_{3,1}, \ldots, \mathsf{Hyb}'_{3,2^{|f_{\mathsf{SMS}}|}}$ and obfuscates $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, C'_{j^*,1})$ (with the punctured key $K[\mathsf{inp}_{j^*}]$ hardwired).

  Indistinguishability of $\mathsf{Hyb}'_{10}(\mathcal{A})$ and $\mathsf{Hyb}'_{11}(\mathcal{A})$ follows by a similar argument as that used in the proof of Lemma 6.10.

- Hyb'_{12} : Same as Hyb'_{11}, except the challenger reverts the change in Hyb'_2 and hardwires the unpunctured key K in the obfuscated circuit (rather than $K[\mathsf{inp}_{j^*}]$).

  As in the $\mathsf{Hyb}'_1 \to \mathsf{Hyb}'_2$ transition, the obfuscated circuit never evaluates the PRF using K at inp_{j^*}, so $\mathsf{Hyb}'_{11}(\mathcal{A})$ and $\mathsf{Hyb}'_{12}(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security (and correctness of the puncturable PRF).

The circuit obfuscated in Hyb'_{12} is functionally equivalent to C'_{j^*+1} (the precomputed tuple has $\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}$ with $x_i = \bot$, so on input inp_{j^*} the circuit outputs what C'_{j^*+1} would output, while on all other inputs the circuits agree). Thus, $\mathsf{Hyb}'_{12}(\mathcal{A})$ and $\mathsf{Hyb}_{4,j^*+1}(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security. We now state and prove Lemma 6.10.

Lemma 6.10. *Suppose $i\mathcal{O}$ is sub-exponentially secure and Π_{PRF} is correct and satisfies pseudorandomness at the punctured point (with sub-exponential security). Then for all $t \in [M'-1]$, hybrids $\mathsf{Hyb}'_{3,t}(\mathcal{A})$ and $\mathsf{Hyb}'_{3,t+1}(\mathcal{A})$ are computationally indistinguishable. Recall that $M' = 2^{|f_{\mathsf{SMS}}|} + 1$.*

Proof. We prove this via a sequence of hybrids.

- $\mathsf{Hyb}'_{3,t,1}$: Same as $\mathsf{Hyb}'_{3,t}$, except the challenger precomputes the output of the obfuscated circuit on input $(f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})$ and hardwires the precomputed value into a modified circuit. Specifically, the challenger computes
$$\overline{d_{\mathsf{SMS},1}} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}[t], \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$$
  and sets
$$\widetilde{d_{\mathsf{SMS},1}} = \overline{d_{\mathsf{SMS},1}} \oplus \mathsf{PRF}(k_1, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})) \oplus \cdots \oplus \mathsf{PRF}(k_n, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})),$$
  where $k_1, \ldots, k_n$ are the keys obtained from the precomputed encoding inp_{j^*}. The challenger now samples $\widetilde{C} \leftarrow i\mathcal{O}(1^{\lambda_2}, \overline{C}_{t,1})$, where $\overline{C}_{t,1}$ is the circuit shown in Fig. 6 below with $\widetilde{d_{\mathsf{SMS},1}}$ added to the hardwired tuple:

Inputs: $(f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n)$.
Hardwired: $K[\mathsf{inp}_{j^*}]$, $\{\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{crs}_{\mathsf{NIZK},j}\}_{j\in[n]}$, $\{\mathsf{sk}_{j,0}\}_{j\neq i}$, $\mathsf{sk}_{i,1}$, $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, (x_i^*, k_i^*), inp_{j^*}, $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$, $f_{\mathsf{SMS}}[t]$, $\widetilde{d_{\mathsf{SMS},1}}$, $\mathsf{crs}_{\mathsf{rSMS}}$, $\mathsf{crs}_{\mathsf{SMS}}$.

1. For every $j \in [n]$, parse pe_j as $(\mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}, \pi_j)$.
2. For every $j \in [n]$,
   (a) Check if $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK},j}, \mathcal{R}_{\mathsf{eq}}, (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \mathsf{ct}_{j,0}, \mathsf{ct}_{j,1}), \pi_j) = 1$.
   (b) If the check fails, output $\perp$.
3. Else, for every $j \in [n]$,
   (a) If $j = i$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, set $(x_i, k_i) = (x_i^*, k_i^*)$.
   (b) Else, if $j = i$, but $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) \neq (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, compute $(x_i, k_i) = \mathsf{PKE.Dec}(\mathsf{sk}_{i,1}, \mathsf{ct}_{i,1})$.
   (c) Else, compute $(x_j, k_j) = \mathsf{PKE.Dec}(\mathsf{sk}_{j,0}, \mathsf{ct}_{j,0})$.
   (d) Compute $\mathsf{mask}_j = \mathsf{PRF.Eval}(k_j, (f_{\mathsf{SMS}}, \mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
4. If $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) = \mathsf{inp}_{j^*}$, then do the following:
   (a) If $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}}) = \perp$, output $\perp$.
   (b) Else if $f_{\mathsf{SMS}} \prec' f_{\mathsf{SMS}}[t]$, set $d_{\mathsf{SMS},1} = 0$.
   (c) Else if $f_{\mathsf{SMS}} = f_{\mathsf{SMS}}[t]$, output $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{d_{\mathsf{SMS},1}})$.
   (d) Otherwise, compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \widetilde{\mathsf{td}_{\mathsf{SMS},1}})$.
   (e) Output $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.
5. Else, compute $(r_{\mathsf{rSMS}}, r_{\mathsf{SMS}}) = \mathsf{PRF.Eval}(K[\mathsf{inp}_{j^*}], (\mathsf{pe}_0, \ldots, \mathsf{pe}_n))$.
6. If $(\mathsf{pe}_0, \ldots, \mathsf{pe}_n) \prec \mathsf{inp}_{j^*}$ and $(\mathsf{ct}_{i,0}, \mathsf{ct}_{i,1}) = (\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$, reset $x_i = \perp$.
7. Compute $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}) = \mathsf{rSMS.Encode}(\mathsf{crs}_{\mathsf{rSMS}}, 1, (x_1, \ldots, x_n); r_{\mathsf{rSMS}})$.
8. Compute $(\mathsf{pe}_{\mathsf{SMS},1}, \mathsf{td}_{\mathsf{SMS},1}) = \mathsf{SMS.Encode}(\mathsf{crs}_{\mathsf{SMS}}, 1, (\mathsf{pe}_0, \mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{td}_{\mathsf{rSMS},1}); r_{\mathsf{SMS}})$.
9. Compute $d_{\mathsf{SMS},1} = \mathsf{SMS.Decode}(1^1, \mathsf{crs}_{\mathsf{SMS}}, 1, f_{\mathsf{SMS}}, \mathsf{td}_{\mathsf{SMS},1})$.
10. Output $(\mathsf{pe}_{\mathsf{rSMS},1}, \mathsf{pe}_{\mathsf{SMS},1}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n)$.

Figure 6: Description of $\overline{C}_{t,1}$.

The functionality of $\overline{C}_{t,1}$ coincides with that of the circuit obfuscated in $\mathsf{Hyb}'_{3,t}$. Specifically, on the input $(f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})$, the circuit $\overline{C}_{t,1}$ outputs the precomputed value $\widetilde{d_{\mathsf{SMS},1}}$:

$$\begin{aligned}\widetilde{d_{\mathsf{SMS},1}} &= \overline{d_{\mathsf{SMS},1}} \oplus \mathsf{PRF}(k_1, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})) \oplus \cdots \oplus \mathsf{PRF}(k_n, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})) \\ &= d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n.\end{aligned}$$

Thus, $\mathsf{Hyb}'_{3,t}(\mathcal{A})$ and $\mathsf{Hyb}'_{3,t,1}(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security.

- $\mathsf{Hyb}'_{3,t,2}$: Same as $\mathsf{Hyb}'_{3,t,1}$, except the challenger samples a punctured key

$$k_i^*[(f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})] \leftarrow \mathsf{PRF.Punc}(k_i^*, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*}))$$

and hardwires this punctured key in place of the unpunctured key k_i^* in the obfuscated circuit.

The modified circuit is functionally equivalent to $\overline{C}_{t,1}$ since the obfuscated circuit never evaluates the PRF using k_i^* at the punctured point $(f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})$. Its output on input $(f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})$ is taken from the precomputed value $\widetilde{d_{\mathsf{SMS},1}}$. On all other points, the punctured and unpunctured keys agree by correctness of Π_{PRF}. Thus, $\mathsf{Hyb}'_{3,t,1}(\mathcal{A})$ and $\mathsf{Hyb}'_{3,t,2}(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security.

- $\mathsf{Hyb}'_{3,t,3}$: Same as $\mathsf{Hyb}'_{3,t,2}$, except the challenger sets the precomputed value to

$$\widetilde{d_{\mathsf{SMS},1}} = 0 \oplus \mathsf{PRF}(k_1, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})) \oplus \cdots \oplus \mathsf{PRF}(k_n, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})).$$

  In other words, the challenger replaces $\overline{d_{\mathsf{SMS},1}}$ with 0.

  By pseudorandomness at the punctured point of Π_{PRF}, the value $\mathsf{PRF}(k_i^*, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*}))$ is pseudorandom given the punctured key $k_i^*[(f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})]$ that is hardwired in the obfuscated circuit. Hence, the hardwired value $\widetilde{d_{\mathsf{SMS},1}}$ is computationally indistinguishable in the two hybrids, and $\mathsf{Hyb}'_{3,t,2}(\mathcal{A})$ and $\mathsf{Hyb}'_{3,t,3}(\mathcal{A})$ are computationally indistinguishable.

- $\mathsf{Hyb}'_{3,t,4}$: Same as $\mathsf{Hyb}'_{3,t,3}$, except the challenger reverts the change in $\mathsf{Hyb}'_{3,t,2}$ and hardwires the unpunctured key k_i^* in the obfuscated circuit (rather than the punctured key).

  By the same argument as the $\mathsf{Hyb}'_{3,t,1} \to \mathsf{Hyb}'_{3,t,2}$ transition, $\mathsf{Hyb}'_{3,t,3}(\mathcal{A})$ and $\mathsf{Hyb}'_{3,t,4}(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security.

Finally, the circuit obfuscated in $\mathsf{Hyb}'_{3,t,4}$ is functionally equivalent to the circuit obfuscated in $\mathsf{Hyb}'_{3,t+1}$. The only (potential) difference between the two circuits is on input $(f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})$. We compare their outputs on this input:

- In $\mathsf{Hyb}'_{3,t,4}$, the circuit outputs $(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, \widetilde{d_{\mathsf{SMS},1}})$ where

$$\widetilde{d_{\mathsf{SMS},1}} = \mathsf{PRF}(k_1, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})) \oplus \cdots \oplus \mathsf{PRF}(k_n, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})).$$

- In $\mathsf{Hyb}'_{3,t+1}$, the condition $f_{\mathsf{SMS}} \prec' f_{\mathsf{SMS}}[t+1]$ holds, so the circuit sets $d_{\mathsf{SMS},1} = 0$ and outputs the tuple

$$(\widetilde{\mathsf{pe}_{\mathsf{rSMS},1}}, \widetilde{\mathsf{pe}_{\mathsf{SMS},1}}, d_{\mathsf{SMS},1} \oplus \mathsf{mask}_1 \oplus \cdots \oplus \mathsf{mask}_n),$$

  where $\mathsf{mask}_j = \mathsf{PRF}(k_j, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*}))$. By construction, the third component thus equals

$$0 \oplus \mathsf{PRF}(k_1, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})) \oplus \cdots \oplus \mathsf{PRF}(k_n, (f_{\mathsf{SMS}}[t], \mathsf{inp}_{j^*})).$$

The two outputs coincide, so the circuits are functionally equivalent. Thus, $\mathsf{Hyb}'_{3,t,4}(\mathcal{A})$ and $\mathsf{Hyb}'_{3,t+1}(\mathcal{A})$ are computationally indistinguishable by $i\mathcal{O}$ security. □

Lemma 6.9 now follows by a hybrid argument. □

As argued earlier, the distribution of $(\mathsf{crs}, \mathsf{pe}_i)$ in Hyb_5 is independent of the input x_i^*. By a hybrid argument, we conclude that $\mathsf{Hyb}_0(\mathcal{A})$ and $\mathsf{Hyb}_5(\mathcal{A})$ are computationally indistinguishable and security holds. □

Theorem 6.11 (Succinctness)**.** *If Π_{rSMS} and Π_{SMS} satisfy succinctness, then Construction 6.5 satisfies succinctness.*

Proof. We consider the size of the public encodings:

- By succinctness of Π_{rSMS}, we have that $|\mathsf{pe}_0| = \mathsf{poly}(\lambda_1, d, \log N) = \mathsf{poly}(\lambda, d, \log N, n, \ell)$.
- Next, for $i \in [n]$, pe_i consists of two encryptions (of size $\mathsf{poly}(\lambda, \ell)$) and a NIZK proof of correctness. We conclude that $|\mathsf{pe}_i| = \mathsf{poly}(\lambda, \ell)$.

Finally, consider the size of the common reference string:

- By succinctness of Π_{rSMS}, we have $|\mathsf{crs}_{\mathsf{rSMS}}| = \mathsf{poly}(\lambda_1, d, \log N, n\ell) = \mathsf{poly}(\lambda, d, \log N, n, \ell)$.
- By succinctness of Π_{SMS}, we have $|\mathsf{crs}_{\mathsf{SMS}}| = \mathsf{poly}(\lambda_1, d, \log N, n\ell) = \mathsf{poly}(\lambda, d, \log N, n, \ell)$.
- Again, by succinctness of Π_{rSMS} and Π_{SMS}, the size of the circuits (appearing in the construction and security analysis) is $\mathsf{poly}(\lambda_2, d, \log N, n\ell) = \mathsf{poly}(\lambda, d, \log N, n, \ell)$.

We conclude that $|\mathsf{crs}| = \mathsf{poly}(\lambda, d, \log N, n, \ell)$. □

7 Leveled Algebraic Homomorphic MAC from LWE

In this section, we show how to construct a leveled algebraic homomorphic MAC that supports bounded-depth Boolean circuits from the plain LWE assumption. This version can be viewed as an adaptation of the algebraic homomorphic MAC based on ring LWE from [ILL25a].

Lattice preliminaries. We start by recalling some preliminaries on lattice-based cryptography. We write $D_{\mathbb{Z},\chi}$ to denote the discrete Gaussian distribution with width parameter $\chi > 0$. We will use the following standard tail bound on the discrete Gaussian distribution (cf. [MP12]):

$$\Pr[|x| > \sqrt{\lambda}\chi : x \leftarrow D_{\mathbb{Z},\chi}] \leq 2^{-\lambda}. \tag{7.1}$$

For a vector $\mathbf{v} \in \mathbb{Z}^n$, we write $\|\mathbf{v}\|$ to denote the ℓ_∞ norm of $\mathbf{v}$. When $\mathbf{v} \in \mathbb{Z}_q^n$, we define $\|\mathbf{v}\|$ to denote the ℓ_∞ norm of the vector obtained by first lifting the components of $\mathbf{v}$ to the integers (with representatives in the interval $(-q/2, q/2]$). For matrices $\mathbf{A}, \mathbf{B}$, we write $\mathbf{A} \otimes \mathbf{B}$ to denote the tensor (Kronecker) product of $\mathbf{A}$ and $\mathbf{B}$. We will use the fact that $\mathbf{A} \otimes \mathbf{B}$ and $\mathbf{B} \otimes \mathbf{A}$ are equivalent up to permutation: namely, for all dimensions n_1, m_1, n_2, m_2 there exist permutation matrices $\mathbf{\Pi}_1 \in \{0,1\}^{n_1 n_2 \times n_1 n_2}$ and $\mathbf{\Pi}_2 \in \{0,1\}^{m_1 m_2 \times m_1 m_2}$ such that for all $\mathbf{A} \in \mathbb{Z}_q^{n_1 \times m_1}$ and $\mathbf{B} \in \mathbb{Z}_q^{n_2 \times m_2}$, we have

$$\mathbf{B} \otimes \mathbf{A} = \mathbf{\Pi}_1 (\mathbf{A} \otimes \mathbf{B}) \mathbf{\Pi}_2.$$

We also use the smudging lemma from [BDE+18]:

Lemma 7.1 (Smudging Lemma [BDE+18, adapted])**.** *Let λ be a security parameter. Take any $e \in \mathbb{Z}$ where $|e| \leq B$. Suppose that $\mathcal{D}$ is either the uniform distribution over $\mathbb{Z}_p$ where $p \geq B \cdot \lambda^{\omega(1)}$ or $\mathcal{D} = D_{\mathbb{Z},\chi}$ where $\chi \geq B \cdot \lambda^{\omega(1)}$. Then, the following distributions are statistically indistinguishable: $\{z : z \leftarrow \mathcal{D}\}$ and $\{z + e : z \leftarrow \mathcal{D}\}$.*

Learning with errors. The learning with errors assumption (in Hermite normal form) $\mathsf{LWE}_{n,m,q,\chi}$ [Reg05, ACPS09] asserts that the following distributions are computationally indistinguishable:

$$(\mathbf{A}, \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal) \quad \text{and} \quad (\mathbf{A}, \mathbf{u}^\intercal) \quad \text{where} \quad \mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m},\ \mathbf{s} \xleftarrow{R} D_{\mathbb{Z},\chi}^n,\ \mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m,\ \mathbf{u} \xleftarrow{R} \mathbb{Z}_q^m.$$

A classic result from [ACPS09] shows that $\mathsf{LWE}_{n,m,q,\chi}$ is essentially equivalent to the more common version of LWE where the secret $\mathbf{s}$ is sampled from the uniform distribution $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$. For our security analysis, we will prove two simple versions that will be useful in our application:

Lemma 7.2 (LWE with Tensored Samples)**.** *Suppose the $\mathsf{LWE}_{n,n^2+n,q,\chi}$ assumption holds. Then, the following two distributions are computationally indistinguishable:*

$$(\mathbf{A},\ \hat{\mathbf{A}},\ (\mathbf{I}_n \otimes \mathbf{s}^\intercal)\mathbf{A} + \mathbf{E},\ \mathbf{s}^\intercal \hat{\mathbf{A}} + \hat{\mathbf{e}}^\intercal) \quad \textit{and} \quad (\mathbf{A},\ \hat{\mathbf{A}},\ \mathbf{U},\ \hat{\mathbf{u}}^\intercal),$$

where $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n^2 \times n}$, $\hat{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$, $\mathbf{s} \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\chi}^{n \times n}$, $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$, and $\hat{\mathbf{u}} \xleftarrow{R} \mathbb{Z}_q^n$.

Proof. Suppose there exists an efficient algorithm $\mathcal{A}$ that can distinguish the two distributions. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for LWE:

- Algorithm $\mathcal{B}$ receives the LWE challenge $(\mathbf{B}, \mathbf{z}^\intercal)$ and parses $\mathbf{B} = [\mathbf{B}_1 \mid \cdots \mid \mathbf{B}_{n+1}]$ where $\mathbf{B}_i \in \mathbb{Z}_q^{n \times n}$ and $\mathbf{z}^\intercal = [\mathbf{z}_1^\intercal \mid \cdots \mid \mathbf{z}_{n+1}^\intercal]$ where $\mathbf{z}_i \in \mathbb{Z}_q^n$.
- Algorithm $\mathcal{B}$ defines

$$\mathbf{A} = \begin{bmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_n \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{A}} = \mathbf{B}_{n+1} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{z}_1^\intercal \\ \vdots \\ \mathbf{z}_n^\intercal \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{y}} = \mathbf{z}_{n+1}.$$

- Algorithm $\mathcal{B}$ runs $\mathcal{A}$ on $(\mathbf{A}, \hat{\mathbf{A}}, \mathbf{Y}, \hat{\mathbf{y}})$ and outputs whatever $\mathcal{A}$ outputs.

Observe that $\mathbf{A}$ and $\hat{\mathbf{A}}$ are uniform over $\mathbb{Z}_q^{n^2 \times n}$ and $\mathbb{Z}_q^{n \times n}$, as required. It suffices to consider the distribution of $\mathbf{Y}$ and $\hat{\mathbf{y}}$:

- Suppose $\mathbf{z}^\intercal = \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}^\intercal$ where $\mathbf{e}^\intercal = [\mathbf{e}_1^\intercal \mid \cdots \mid \mathbf{e}_{n+1}^\intercal]$. In this case, we can write

$$\mathbf{Y} = \begin{bmatrix} \mathbf{z}_1^\intercal \\ \vdots \\ \mathbf{z}_n^\intercal \end{bmatrix} = \begin{bmatrix} \mathbf{s}^\intercal \mathbf{B}_1 + \mathbf{e}_1^\intercal \\ \vdots \\ \mathbf{s}^\intercal \mathbf{B}_n + \mathbf{e}_n^\intercal \end{bmatrix} = (\mathbf{I}_n \otimes \mathbf{s}^\intercal) \begin{bmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_n \end{bmatrix} + \begin{bmatrix} \mathbf{e}_1^\intercal \\ \vdots \\ \mathbf{e}_n^\intercal \end{bmatrix} = (\mathbf{I}_n \otimes \mathbf{s}^\intercal)\mathbf{A} + \mathbf{E} \quad \text{where} \quad \mathbf{E} = \begin{bmatrix} \mathbf{e}_1^\intercal \\ \vdots \\ \mathbf{e}_n^\intercal \end{bmatrix}.$$

  Finally, $\hat{\mathbf{y}}^\intercal = \mathbf{z}_{n+1}^\intercal = \mathbf{s}^\intercal \mathbf{B}_{n+1} + \mathbf{e}_{n+1}^\intercal = \mathbf{s}^\intercal \hat{\mathbf{A}} + \mathbf{e}_{n+1}^\intercal$. This corresponds to the pseudorandom distribution in the lemma.
- Suppose $\mathbf{z} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n^2+n}$. In this case, the distribution of $\mathbf{Y}$ is uniform over $\mathbb{Z}_q^{n \times n}$ and the distribution of $\hat{\mathbf{y}}$ is uniform over $\mathbb{Z}_q^n$. This corresponds to the random distribution.

We conclude that the distinguishing advantage of $\mathcal{B}$ is precisely the same as the distinguishing advantage of $\mathcal{A}$. The claim holds. □

Corollary 7.3 (LWE with Tensored Samples)**.** *Suppose the* $\mathsf{LWE}_{n,n^2+n,q,\chi}$ *assumption holds. Then, the following two distributions are computationally indistinguishable:*

$$(\mathbf{A}\,,\ \hat{\mathbf{A}}\,,\ (\mathbf{s}^\intercal \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}\,,\ \mathbf{s}^\intercal \hat{\mathbf{A}} + \hat{\mathbf{e}}^\intercal) \quad \textit{and} \quad (\mathbf{A}\,,\ \hat{\mathbf{A}}\,,\ \mathbf{U}\,,\ \hat{\mathbf{u}}^\intercal),$$

where $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n^2 \times n}$, $\hat{\mathbf{A}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$, $\mathbf{s} \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\chi}^{n \times n}$, $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$, *and* $\hat{\mathbf{u}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$.

Proof. Let $\mathbf{\Pi}_1 \in \{0,1\}^{n^2 \times n^2}$ and $\mathbf{\Pi}_2 \in \{0,1\}^{n \times n}$ be the permutation matrices such that for all $\mathbf{s} \in \mathbb{Z}_q^n$

$$(\mathbf{I}_n \otimes \mathbf{s}^\intercal) = \mathbf{\Pi}_1 (\mathbf{s}^\intercal \otimes \mathbf{I}_n) \mathbf{\Pi}_2. \tag{7.2}$$

We now proceed via a hybrid argument. Specifically, consider the following distributions:

- $\mathcal{D}_0$: Sample $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n^2 \times n}$, $\hat{\mathbf{A}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$, $\mathbf{s} \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\chi}^{n \times n}$, $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\chi}^n$. Output

$$(\mathbf{A}\,,\ \hat{\mathbf{A}}\,,\ (\mathbf{s}^\intercal \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}\,,\ \mathbf{s}^\intercal \hat{\mathbf{A}} + \hat{\mathbf{e}}^\intercal).$$

- $\mathcal{D}_1$: Sample $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n^2 \times n}$, $\hat{\mathbf{A}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$, $\mathbf{s} \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\chi}^{n \times n}$, $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\chi}^n$. Output

$$(\mathbf{\Pi}_2 \mathbf{A}\,,\ \hat{\mathbf{A}}\,,\ \mathbf{\Pi}_1^{-1} \mathbf{\Pi}_1 (\mathbf{s}^\intercal \otimes \mathbf{I}_n) \mathbf{\Pi}_2 \mathbf{A} + \mathbf{\Pi}_1^{-1} \mathbf{E}\,,\ \mathbf{s}^\intercal \hat{\mathbf{A}} + \hat{\mathbf{e}}^\intercal).$$

  This is identical to the previous distribution since $\mathbf{\Pi}_1, \mathbf{\Pi}_2$ are *permutation* matrices.
- $\mathcal{D}_2$: Sample $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n^2 \times n}$, $\hat{\mathbf{A}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$, $\mathbf{s} \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\chi}^{n \times n}$, $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\chi}^n$. Output

$$(\mathbf{\Pi}_2 \mathbf{A}\,,\ \hat{\mathbf{A}}\,,\ \mathbf{\Pi}_1^{-1} (\mathbf{I}_n \otimes \mathbf{s}^\intercal) \mathbf{A} + \mathbf{\Pi}_1^{-1} \mathbf{E}\,,\ \mathbf{s}^\intercal \hat{\mathbf{A}} + \hat{\mathbf{e}}^\intercal).$$

  This is identical to the previous distribution by Eq. (7.2).

- $\mathcal{D}_3$: Sample $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n^2 \times n}$, $\hat{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$, $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$, and $\hat{\mathbf{u}} \xleftarrow{R} \mathbb{Z}_q^n$. Output

$$(\boldsymbol{\Pi}_2 \mathbf{A}\ ,\ \hat{\mathbf{A}}\ ,\ \boldsymbol{\Pi}_1^{-1}\mathbf{U}\ ,\ \hat{\mathbf{u}}^\intercal).$$

  This is computationally indistinguishable from $\mathcal{D}_2$ under $\mathsf{LWE}_{n,n^2+n,q,\chi}$ by Lemma 7.2.

- $\mathcal{D}_4$: Sample $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n^2 \times n}$, $\hat{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$, $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$, and $\hat{\mathbf{u}} \xleftarrow{R} \mathbb{Z}_q^n$. Output

$$(\mathbf{A}\ ,\ \hat{\mathbf{A}}\ ,\ \mathbf{U}\ ,\ \hat{\mathbf{u}}^\intercal).$$

  This is identical to the previous distribution since $\boldsymbol{\Pi}_1$ and $\boldsymbol{\Pi}_2$ are permutation matrices.

By a hybrid argument, we conclude that $\mathcal{D}_0$ and $\mathcal{D}_4$ are computationally indistinguishable and the claim holds. □

Construction 7.4 (Leveled Algebraic Homomorphic MAC from LWE)**.** Let λ be a security parameter and $n = n(\lambda)$ be the lattice dimension. Let $p = p(\lambda), q = q(\lambda)$ be moduli where $q = \alpha p$ for some $\alpha \in \mathbb{N}$. Let $\chi = \chi(\lambda)$ be a width parameter and $\tilde{\chi} = \tilde{\chi}(\lambda)$ be a width parameter (for smudging). Let $F\colon \mathcal{K} \times \{0,1\}^{\lambda+1} \to \mathbb{Z}_q^n$ be a PRF; we will interpret the domain of F as a pair (t, b) where $t \in [2^\lambda]$ and $b \in \{0,1\}$. For ease of exposition, we assume that the circuits C in this construction have the following structure:

- We assume that each gate in the Boolean circuit is either a binary multiplication gate or a unary negation gate. This is sufficient to implement a NAND gate. We assume that the only inputs to negation gates are either input wires or the output of a multiplication gate.
- Let w be the number of wires in $C\colon \{0,1\}^\ell \to \{0,1\}$. We order the wires according to a canonical topological order where the first ℓ wires correspond to the input and wire w corresponds to the output. We associate with each wire $i \in [w]$ an encoding depth d_i. The requirement is that the encoding depth associated with the input wires to any gate in the circuit is the same.
- We define the encoding depth of the input wires $i \in [\ell]$ to be $d_i = 0$.
- For an intermediate wire $t \in [w]$ in the circuit
  - If the gate is a negation gate with input wire $i \in [w]$, then define $d_t = d_i$.
  - If the gate is a multiplication gate with input wires $i, j \in [w]$ and $d_i = d_j$, then $d_t = d_i + 1 = d_j + 1$.

We can generically transform any Boolean circuit C (consisting of multiplication gates and negation gates) with depth d into a functionally-equivalent circuit C' with the following two properties:

- C' has the same depth d and the size of C' is larger than that of C by a multiplicative factor $O(d)$.
- The encoding depth associated with the input wires to each multiplication gate in C' is the same.

The approach is to "lift" a wire with encoding depth d_i into a wire with encoding depth d_{i+1} by multiplying it with itself (since $b^2 = b$ for all $b \in \{0,1\}$). Thus, whenever a multiplication gate in circuit C operates on two wires with different encoding depths, we first lift the wire with the lower encoding depth to the encoding depth of the other wire using repeated squaring. At most $O(d)$ such operations are needed. Thus, in the following description, we will assume all of the Boolean circuits C satisfy this property. We now construct a leveled algebraic homomorphic MAC $\Pi_{\mathsf{AHMAC}} = (\mathsf{Setup}, \mathsf{EvalKey}, \mathsf{EvalTag})$ over $\mathbb{Z}_p^n$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^d, \mathbf{s})$: On input the security parameter λ, the depth bound d, and a global offset $\mathbf{s} \in \mathbb{Z}_p^n$, the setup algorithm proceeds as follows:
  - Sample a PRF key $k \xleftarrow{R} \mathcal{K}$.
  - Sample the MAC key associated with input wires $\mathbf{s}_{\mathsf{in}} \leftarrow D_{\mathbb{Z},\chi}^{\boldsymbol{n}}$.

- For each $i \in [d-1]$, sample $\mathbf{s}_i, \hat{\mathbf{s}}_i \leftarrow D^n_{\mathbb{Z},\chi}$. These can be viewed as MAC keys associated with level i of the circuit. Let $\mathbf{s}_0 = \mathbf{s}_{\mathsf{in}}$ and $\mathbf{s}_d = \mathbf{s}$.
- Sample a matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n^2 \times n}$, and for each $i \in [0, d-1]$, sample $\mathbf{E}_{i,1}, \mathbf{E}_{i,2} \leftarrow D^{n\times n}_{\mathbb{Z},\chi}$ and $\tilde{\mathbf{e}}_{i,3} \leftarrow D^n_{\mathbb{Z},\tilde{\chi}}$. Then compute the evaluation matrices

$$\begin{aligned}\mathbf{B}_{i,1} &= (\mathbf{I}_n \otimes \hat{\mathbf{s}}_i^\intercal)\mathbf{A} + \mathbf{E}_{i,1} \in \mathbb{Z}_q^{n\times n}\\ \mathbf{B}_{i,2} &= (\mathbf{s}_i^\intercal \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}_{i,2} \in \mathbb{Z}_q^{n\times n}\\ \mathbf{b}_{i,3}^\intercal &= (\mathbf{s}_i^\intercal \otimes \hat{\mathbf{s}}_i^\intercal)\mathbf{A} + \alpha \mathbf{s}_{i+1}^\intercal + \tilde{\mathbf{e}}_{i,3}^\intercal \in \mathbb{Z}_q^n.\end{aligned}$$

- Next, sample $\hat{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n\times n}$. Then, for each $i \in [0, d-1]$, sample $\hat{\mathbf{e}}_i \leftarrow D^n_{\mathbb{Z},\chi}$ and construct the translation vector

$$\mathbf{t}_i^\intercal = \mathbf{s}_i^\intercal \hat{\mathbf{A}} + \hat{\mathbf{e}}_i^\intercal - \alpha \hat{\mathbf{s}}_i^\intercal \in \mathbb{Z}_q^n.$$

- For each $i \in [0, d-1]$, sample $\hat{\mathbf{k}}_i \xleftarrow{R} \mathbb{Z}_p^n$ and compute $\hat{\boldsymbol{\sigma}}_i = \mathbf{s}_i + \hat{\mathbf{k}}_i \in \mathbb{Z}_p^n$.

Output the input MAC key $\mathbf{s}_{\mathsf{in}}$, the secret key $\mathsf{sk} = (\hat{\mathbf{k}}_0, \ldots, \hat{\mathbf{k}}_{d-1})$, and the evaluation key evk defined as follows:

$$\mathsf{evk} = \left(k, \mathbf{A}, \hat{\mathbf{A}}, \left\{\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \mathbf{b}_{i,3}, \mathbf{t}_i, \hat{\boldsymbol{\sigma}}_i\right\}_{i\in[0,d-1]}\right). \tag{7.3}$$

- $\mathsf{EvalKey}(\mathsf{sk}, \mathsf{evk}, C, (\bar{\mathbf{k}}_1, \ldots, \bar{\mathbf{k}}_\ell))$: On input the secret key $\mathsf{sk} = (\hat{\mathbf{k}}_0, \ldots, \hat{\mathbf{k}}_{d-1})$ and the evaluation key evk (whose components are parsed according to Eq. (7.3)), a Boolean circuit $C\colon \{0,1\}^\ell \to \{0,1\}$, and input keys $\bar{\mathbf{k}}_1, \ldots, \bar{\mathbf{k}}_\ell \in \mathbb{Z}_p^n$, the key-evaluation algorithm proceeds as follows:
  - Let w be the number of wires in C and number them according to their canonical topological order. We will associate a pair $(\mathbf{k}_i, d_i)$ with each wire, where $\mathbf{k}_i$ denotes the key associated with wire i and d_i denotes the encoding depth (where the encoding depth is as defined above).
  - For each input wire, let $\boldsymbol{\delta}_{i,1} = F(k, (i,1))$ and $\mathbf{k}_i = \bar{\mathbf{k}}_i + \lfloor \boldsymbol{\delta}_{i,1}/\alpha \rceil \in \mathbb{Z}_p^n$.
  - For each gate in the circuit (considered in topological order) with output wire $t \in [w]$, compute the key $\mathbf{k}_t$ associated with the output wire as follows:
    - If the gate is a negation gate with input wire $i \in [w]$, compute $\boldsymbol{\delta}_{t,1} = F(k, (t,1))$ and let $\mathbf{k}_t = \hat{\mathbf{k}}_{d_i} - \mathbf{k}_i + \lfloor \boldsymbol{\delta}_{t,1}/\alpha \rceil \in \mathbb{Z}_p^n$.
    - If the gate is a multiplication gate with input wires $i, j \in [w]$, compute shifts $\boldsymbol{\delta}_{t,0} = F(k, (t,0))$ and $\boldsymbol{\delta}_{t,1} = F(k, (t,1))$. Then, set

$$\begin{aligned}\hat{\mathbf{k}}_j^\intercal &= \lfloor (\mathbf{k}_j^\intercal \hat{\mathbf{A}} + \boldsymbol{\delta}_{t,0}^\intercal)/\alpha \rceil \in \mathbb{Z}_p^n\\ \mathbf{k}_t^\intercal &= \lfloor ((\mathbf{k}_i^\intercal \otimes \hat{\mathbf{k}}_j^\intercal)\mathbf{A} + \boldsymbol{\delta}_{t,1}^\intercal)/\alpha \rceil \in \mathbb{Z}_p^n.\end{aligned}$$

    The output is $\mathbf{k}_w$ (i.e., the key associated with the output wire).

- $\mathsf{EvalTag}(\mathsf{evk}, C, \mathbf{x}, \boldsymbol{\sigma}_{\mathbf{x}})$: On input the evaluation key evk (whose components are parsed according to Eq. (7.3)), a Boolean circuit $C\colon \{0,1\}^\ell \to \{0,1\}$, an input $\mathbf{x} \in \{0,1\}^\ell$ and tags $\bar{\boldsymbol{\sigma}}_{\mathbf{x}} = (\bar{\boldsymbol{\sigma}}_1, \ldots, \bar{\boldsymbol{\sigma}}_\ell)$, the tag-evaluation algorithm proceeds as follows:
  - Let w be the number of wires in C and number them according to their canonical topological order. We will associate a pair $(\boldsymbol{\sigma}_i, d_i)$ with each wire, where $\boldsymbol{\sigma}_i$ denotes the tag associated with wire i and d_i denotes the encoding depth (where the encoding depth is as defined above).
  - Compute $C(\mathbf{x})$ and let $\beta_i \in \{0,1\}$ be the value of wire i in $C(\mathbf{x})$.
  - For each input wire $i \in [\ell]$, let $\boldsymbol{\delta}_{i,1} = F(k, (i,1))$ and $\boldsymbol{\sigma}_i = \bar{\boldsymbol{\sigma}}_i + \lfloor \boldsymbol{\delta}_{i,1}/\alpha \rceil \in \mathbb{Z}_p^n$.

– For each gate in the circuit (considered in topological order) with output wire $t \in [w]$, compute the tag $\boldsymbol{\sigma}_t$ associated with the output wire as follows:
  * If the gate is a negation gate with input wire $i \in [w]$, compute $\boldsymbol{\delta}_{t,1} = F(k,(t,1))$ and let $\boldsymbol{\sigma}_t = \hat{\boldsymbol{\sigma}}_{d_i} - \boldsymbol{\sigma}_i + \lfloor \boldsymbol{\delta}_{t,1}/\alpha \rceil \in \mathbb{Z}_p^n$.
  * If the gate is a multiplication gate with input wires $i, j \in [w]$, compute shifts $\boldsymbol{\delta}_{t,0} = F(k,(t,0))$ and $\boldsymbol{\delta}_{t,1} = F(k,(t,1))$. Recall that by assumption, $d_i = d_j$. Then, compute

$$\hat{\boldsymbol{\sigma}}_j^\intercal = \lfloor (\boldsymbol{\sigma}_j^\intercal \hat{\mathbf{A}} - \beta_j \mathbf{t}_{d_i}^\intercal + \boldsymbol{\delta}_{t,0}^\intercal)/\alpha \rceil \in \mathbb{Z}_p^n.$$

    Then it computes

$$\tilde{\boldsymbol{\sigma}}_t^\intercal = (\boldsymbol{\sigma}_i^\intercal \otimes \hat{\boldsymbol{\sigma}}_j^\intercal)\mathbf{A} - \beta_j \boldsymbol{\sigma}_i^\intercal \mathbf{B}_{d_i,1} - \beta_i \hat{\boldsymbol{\sigma}}_j^\intercal \mathbf{B}_{d_i,2} + \beta_i \beta_j \mathbf{b}_{d_i,3}^\intercal \in \mathbb{Z}_q^n.$$

    Finally, it sets $\boldsymbol{\sigma}_t = \lfloor (\tilde{\boldsymbol{\sigma}}_t + \boldsymbol{\delta}_{t,1})/\alpha \rceil \in \mathbb{Z}_p^n$.

  The output is $\boldsymbol{\sigma}_w$ (i.e., the tag associated with the output wire).

Theorem 7.5 (Correctness). *Suppose $n = \mathsf{poly}(\lambda)$, $p > \lambda^{\omega(1)} \cdot \chi$, and $\alpha > \lambda^{\omega(1)} \cdot p \cdot \max(\chi, \tilde{\chi})$. Suppose F is a secure PRF. Then Construction 7.4 is correct.*

Proof. Take any $\lambda, d \in \mathbb{N}$, global offset $\mathbf{s} \in \mathbb{Z}_p^n$, Boolean circuit $C\colon \{0,1\}^\ell \to \{0,1\}$ of size at most $\mathsf{poly}(\lambda)$ and depth at most d, input $\mathbf{x} \in \{0,1\}^\ell$ and keys $\bar{\mathbf{k}}_1, \ldots, \bar{\mathbf{k}}_\ell \in \mathbb{Z}_p^n$. Let

$$\begin{aligned}
(\mathbf{s}_{\mathsf{in}}, \mathsf{sk}, \mathsf{evk}) &\leftarrow \mathsf{Setup}(1^\lambda, 1^d, \mathbf{s}) \\
\forall i \in [\ell] : \bar{\boldsymbol{\sigma}}_i &= x_i \cdot \mathbf{s}_{\mathsf{in}} + \bar{\mathbf{k}}_i \\
\mathbf{k}_C &= \mathsf{EvalKey}(\mathsf{sk}, \mathsf{evk}, C, (\bar{\mathbf{k}}_1, \ldots, \bar{\mathbf{k}}_\ell)) \\
\boldsymbol{\sigma}' &= \mathsf{EvalTag}(\mathsf{evk}, C, \mathbf{x}, (\bar{\boldsymbol{\sigma}}_1, \ldots, \bar{\boldsymbol{\sigma}}_\ell)).
\end{aligned}$$

We need to show that with overwhelming probability, $\boldsymbol{\sigma}' = C(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_C$. By construction, EvalKey and EvalTag compute the shifts $\boldsymbol{\delta}_{t,i}$ by evaluating $F(k,(t,i))$, where $k \xleftarrow{R} \mathcal{K}$ is the key sampled during Setup. To show the claim, we proceed in two steps:

- First, suppose we evaluated EvalKey and EvalTag using a modified procedure where the shifts are derived as $\boldsymbol{\delta}_{t,i} = f(t,i)$ and $f\colon \{0,1\}^{\lambda+1} \to \mathbb{Z}_q^n$ is a uniform random function. We show that in this setting, with overwhelming probability over the choice of f, it will be the case that $\boldsymbol{\sigma}' = C(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_C$ (over the integers).
- Since correctness holds when the shifts $\boldsymbol{\delta}_{t,i}$ are sampled uniformly at random from $\mathbb{Z}_q^n$ and F is a secure PRF, the same property holds when we derive the shifts as outputs of the PRF. This is because computing EvalKey and EvalTag is an efficient computation when the circuit size and circuit depth are polynomially-bounded, as is checking whether $\boldsymbol{\sigma}' = C(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_C$. Thus, the correctness probability can decrease by at most a negligible amount when the $\boldsymbol{\delta}_{t,i}$ are derived from the PRF.

Thus, it suffices to show that correctness holds when $\boldsymbol{\delta}_{t,i} \xleftarrow{R} \mathbb{Z}_q^n$. To do so, we show inductively that the following invariant holds with overwhelming probability:

- Let w be the number of wires in C and number them according to their canonical topological order (as in EvalKey and EvalTag). Let $(\mathbf{k}_i, \boldsymbol{\sigma}_i, d_i)$ be the key, tag, and encoding depth associated with wire i as computed by EvalKey and EvalTag. Let $\beta_i \in \{0,1\}$ be the value of wire i in $C(\mathbf{x})$.
- The following invariant holds for all wires $i \in [w]$:

$$\boldsymbol{\sigma}_i = \beta_i \cdot \mathbf{s}_{d_i} + \mathbf{k}_i \in \mathbb{Z}^n \tag{7.4}$$

  where $\mathbf{s}_0 = \mathbf{s}_{\mathsf{in}}$ and $\mathbf{s}_d = \mathbf{s}$. Note that this is a relation over the *integers*.

Observe that if $i = w$, Eq. (7.4) says that $\boldsymbol{\sigma}_w = C(\mathbf{x}) \cdot \mathbf{s} + \mathbf{k}_w$. Since EvalKey outputs $\mathbf{k}_w$ and EvalTag outputs $\boldsymbol{\sigma}_w$, correctness holds.

Base case. Consider the input wires $i \in [\ell]$. By definition,

$$\boldsymbol{\sigma}_i = \bar{\boldsymbol{\sigma}}_i + \lfloor \boldsymbol{\delta}_{i,1}/\alpha \rceil = x_i \cdot \mathbf{s}_{\mathsf{in}} + \bar{\mathbf{k}}_i + \lfloor \boldsymbol{\delta}_{i,1}/\alpha \rceil = x_i \cdot \mathbf{s}_{\mathsf{in}} + \mathbf{k}_i \in \mathbb{Z}_p^n.$$

This is a relation modulo p. We need to show that it also holds over the integers. To see this, consider the distribution of $\mathbf{k}_i$. Since $\boldsymbol{\delta}_{i,1} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $q = \alpha p$, the distribution of $\lfloor \boldsymbol{\delta}_{i,1}/\alpha \rceil$ is uniform over $\mathbb{Z}_p^n$. We conclude that $\mathbf{k}_i = \bar{\mathbf{k}}_i + \lfloor \boldsymbol{\delta}_{i,1}/\alpha \rceil$ is uniform over $\mathbb{Z}_p^n$. Next the Setup algorithm samples $\mathbf{s}_{\mathsf{in}} \leftarrow D_{\mathbb{Z},\chi}^n$, so with overwhelming probability, $\|\mathbf{s}_{\mathsf{in}}\| \le \sqrt{\lambda}\chi$. Since $p > \lambda^{\omega(1)} \cdot \chi$, with overwhelming probability over the choice of $\boldsymbol{\delta}_{i,1}$, it holds that $\|\mathbf{k}_i \bmod p\| < p/2 - \|\mathbf{s}_{\mathsf{in}} \bmod p\|$. In this case, $x_i \cdot \mathbf{s}_{\mathsf{in}} + \mathbf{k}_i = (x_i \cdot \mathbf{s}_{\mathsf{in}} + \mathbf{k}_i \bmod p)$, and Eq. (7.4) holds over the integers.

Inductive step. Consider an intermediate wire $t \in [w]$ that is the output of some gate in C. We consider two cases:

- Suppose t is the output of a negation gate with input wire $i \in [w]$. In this case, $\beta_t = 1 - \beta_i$ and $d_t = d_i$. By the inductive hypothesis, this means $\boldsymbol{\sigma}_i = \beta_i \cdot \mathbf{s}_{d_i} + \mathbf{k}_i \in \mathbb{Z}^n$. By construction, EvalKey and EvalTag set

$$\mathbf{k}_t = \hat{\mathbf{k}}_{d_i} - \mathbf{k}_i + \lfloor \boldsymbol{\delta}_{t,1}/\alpha \rceil \in \mathbb{Z}_p^n$$
$$\boldsymbol{\sigma}_t = \hat{\boldsymbol{\sigma}}_{d_i} - \boldsymbol{\sigma}_i + \lfloor \boldsymbol{\delta}_{t,1}/\alpha \rceil \in \mathbb{Z}_p^n.$$

  Finally, the Setup algorithm defines $\hat{\boldsymbol{\sigma}}_{d_i} \coloneqq \mathbf{s}_{d_i} + \hat{\mathbf{k}}_{d_i} \in \mathbb{Z}_p^n$. Putting everything together, we have

$$\begin{aligned}\boldsymbol{\sigma}_t &= \hat{\boldsymbol{\sigma}}_{d_i} - \boldsymbol{\sigma}_i + \lfloor \boldsymbol{\delta}_{t,1}/\alpha \rceil \\ &= \left(\mathbf{s}_{d_i} + \hat{\mathbf{k}}_{d_i}\right) - \left(\beta_i \mathbf{s}_{d_i} + \mathbf{k}_i\right) + \lfloor \boldsymbol{\delta}_{t,1}/\alpha \rceil \\ &= (1-\beta_i)\mathbf{s}_{d_i} + \hat{\mathbf{k}}_{d_i} - \mathbf{k}_i + \lfloor \boldsymbol{\delta}_{t,1}/\alpha \rceil \\ &= \beta_t \cdot \mathbf{s}_{d_t} + \mathbf{k}_t \in \mathbb{Z}_p^n.\end{aligned}$$

  We now need to argue that this relation holds over the integers. This follows via the same argument as in the base case. Namely, over the choice of $\boldsymbol{\delta}_{t,1} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, the distribution of $\mathbf{k}_t$ is uniform over $\mathbb{Z}_p^n$. Since $\|\mathbf{s}_{d_t}\| \le \sqrt{\lambda}\chi$ and $p > \lambda^{\omega(1)} \cdot \chi$, Eq. (7.4) holds with overwhelming probability.

- Suppose t is the output of a multiplication gate with input wires $i, j \in [w]$. In this case, $\beta_t = \beta_i \beta_j$ and $d_i = d_j$ and $d_t = d_i + 1$. By the inductive hypothesis, this means

$$\boldsymbol{\sigma}_i = \beta_i \cdot \mathbf{s}_{d_i} + \mathbf{k}_i \in \mathbb{Z}^n \quad \text{and} \quad \boldsymbol{\sigma}_j = \beta_j \cdot \mathbf{s}_{d_i} + \mathbf{k}_j \in \mathbb{Z}^n.$$

  Consider the tag $\hat{\boldsymbol{\sigma}}_j$ and key $\hat{\mathbf{k}}_j$ computed by EvalTag and EvalKey, respectively. By definition,

$$\hat{\boldsymbol{\sigma}}_j^{\mathsf{T}} = \lfloor (\boldsymbol{\sigma}_j^{\mathsf{T}} \hat{\mathbf{A}} - \beta_j \mathbf{t}_{d_i}^{\mathsf{T}} + \boldsymbol{\delta}_{t,0}^{\mathsf{T}})/\alpha \rceil \in \mathbb{Z}_p^n.$$

  Substituting the relations for $\boldsymbol{\sigma}_j$ and $\mathbf{t}_{d_i}$ (from Setup), the following holds (over $\mathbb{Z}_q^n$):

$$\begin{aligned}\boldsymbol{\sigma}_j^{\mathsf{T}} \hat{\mathbf{A}} - \beta_j \mathbf{t}_{d_i}^{\mathsf{T}} + \boldsymbol{\delta}_{t,0}^{\mathsf{T}} &= \beta_j \mathbf{s}_{d_i}^{\mathsf{T}} \hat{\mathbf{A}} + \mathbf{k}_j^{\mathsf{T}} \hat{\mathbf{A}} - \beta_j (\mathbf{s}_{d_i}^{\mathsf{T}} \hat{\mathbf{A}} + \hat{\mathbf{e}}_{d_i}^{\mathsf{T}} - \alpha \hat{\mathbf{s}}_{d_i}^{\mathsf{T}}) + \boldsymbol{\delta}_{t,0}^{\mathsf{T}} \\ &= \mathbf{k}_j^{\mathsf{T}} \hat{\mathbf{A}} + \alpha \beta_j \hat{\mathbf{s}}_{d_i}^{\mathsf{T}} - \beta_j \hat{\mathbf{e}}_{d_i}^{\mathsf{T}} + \boldsymbol{\delta}_{t,0}^{\mathsf{T}} \in \mathbb{Z}_q^n.\end{aligned}$$

  By Eq. (7.1) and a union bound, with overwhelming probability, we have $\|\hat{\mathbf{e}}_{d_i}\| \le \sqrt{\lambda}\chi$. Since $p \cdot \|\hat{\mathbf{e}}_{d_i}\|/q \le \sqrt{\lambda}\chi/\alpha = \mathsf{negl}(\lambda)$, by Lemma 3.1, we have with overwhelming probability over the choice of $\boldsymbol{\delta}_{t,0} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$,

$$\begin{aligned}\hat{\boldsymbol{\sigma}}_j^{\mathsf{T}} = \lfloor (\boldsymbol{\sigma}_j^{\mathsf{T}} \hat{\mathbf{A}} - \beta_j \mathbf{t}_{d_i}^{\mathsf{T}} + \boldsymbol{\delta}_{t,0}^{\mathsf{T}})/\alpha \rceil &= \lfloor (\mathbf{k}_j^{\mathsf{T}} \hat{\mathbf{A}} + \alpha \beta_j \hat{\mathbf{s}}_{d_i}^{\mathsf{T}} - \beta_j \hat{\mathbf{e}}_{d_i}^{\mathsf{T}} + \boldsymbol{\delta}_{t,0}^{\mathsf{T}})/\alpha \rceil \\ &= \lfloor (\mathbf{k}_j^{\mathsf{T}} \hat{\mathbf{A}} + \alpha \beta_j \hat{\mathbf{s}}_{d_i}^{\mathsf{T}} + \boldsymbol{\delta}_{t,0}^{\mathsf{T}})/\alpha \rceil \\ &= \beta_j \cdot \hat{\mathbf{s}}_{d_i}^{\mathsf{T}} + \lfloor (\mathbf{k}_j^{\mathsf{T}} \hat{\mathbf{A}} + \boldsymbol{\delta}_{t,0}^{\mathsf{T}})/\alpha \rceil \\ &= \beta_j \cdot \hat{\mathbf{s}}_{d_i}^{\mathsf{T}} + \hat{\mathbf{k}}_j \in \mathbb{Z}_p^n.\end{aligned}$$

By the same reasoning as in the base case, over the randomness of $\boldsymbol{\delta}_{t,0} \xleftarrow{R} \mathbb{Z}_q^n$, the distribution of $\hat{\mathbf{k}}_j$ is uniform over $\mathbb{Z}_p^n$. Since $\|\hat{\mathbf{s}}_{d_i}\| \le \sqrt{\lambda}\chi$, and $p > \lambda^{\omega(1)} \cdot \chi$, this means the above relation also holds over the integers with overwhelming probability. Next, we consider the terms that comprise $\tilde{\boldsymbol{\sigma}}_t$. In the following, the operations are taken over $\mathbb{Z}_q^n$, but note that the relationships involving the tags $\boldsymbol{\sigma}_i, \hat{\boldsymbol{\sigma}}_j$ hold over the *integers* (and thus, also over $\mathbb{Z}_q^n$):

$$\begin{aligned}
(\boldsymbol{\sigma}_i^\intercal \otimes \hat{\boldsymbol{\sigma}}_j^\intercal)\mathbf{A} &= \left(\left(\beta_i \mathbf{s}_{d_i}^\intercal + \mathbf{k}_i^\intercal\right) \otimes \left(\beta_j \hat{\mathbf{s}}_{d_i}^\intercal + \hat{\mathbf{k}}_j^\intercal\right)\right)\mathbf{A} \\
&= \beta_i\beta_j\left(\mathbf{s}_{d_i}^\intercal \otimes \hat{\mathbf{s}}_{d_i}^\intercal\right)\mathbf{A} + \beta_i\left(\mathbf{s}_{d_i}^\intercal \otimes \hat{\mathbf{k}}_j^\intercal\right)\mathbf{A} + \beta_j\left(\mathbf{k}_i^\intercal \otimes \hat{\mathbf{s}}_{d_i}^\intercal\right)\mathbf{A} + (\mathbf{k}_i^\intercal \otimes \hat{\mathbf{k}}_j^\intercal)\mathbf{A} \\
\beta_j\boldsymbol{\sigma}_i^\intercal\mathbf{B}_{d_i,1} &= \beta_j\boldsymbol{\sigma}_i^\intercal\left((\mathbf{I}_n \otimes \hat{\mathbf{s}}_{d_i}^\intercal)\mathbf{A} + \mathbf{E}_{d_i,1}\right) \\
&= \beta_j(\beta_i\mathbf{s}_{d_i}^\intercal + \mathbf{k}_i^\intercal)(\mathbf{I}_n \otimes \hat{\mathbf{s}}_{d_i}^\intercal)\mathbf{A} + \beta_j\boldsymbol{\sigma}_i^\intercal\mathbf{E}_{d_i,1} \\
&= \beta_i\beta_j\left(\mathbf{s}_{d_i}^\intercal \otimes \hat{\mathbf{s}}_{d_i}^\intercal\right)\mathbf{A} + \beta_j\left(\mathbf{k}_i^\intercal \otimes \hat{\mathbf{s}}_{d_i}^\intercal\right)\mathbf{A} + \beta_j\boldsymbol{\sigma}_i^\intercal\mathbf{E}_{d_i,1} \\
\beta_i\hat{\boldsymbol{\sigma}}_j^\intercal\mathbf{B}_{d_i,2} &= \beta_i\hat{\boldsymbol{\sigma}}_j^\intercal\left((\mathbf{s}_{d_i}^\intercal \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}_{d_i,2}\right) \\
&= \beta_i\beta_j\left(\mathbf{s}_{d_i}^\intercal \otimes \hat{\mathbf{s}}_{d_i}^\intercal\right)\mathbf{A} + \beta_i\left(\mathbf{s}_{d_i}^\intercal \otimes \hat{\mathbf{k}}_j^\intercal\right)\mathbf{A} + \beta_i\hat{\boldsymbol{\sigma}}_j^\intercal\mathbf{E}_{d_i,2} \\
\beta_i\beta_j\mathbf{b}_{d_i,3}^\intercal &= \beta_i\beta_j\left(\mathbf{s}_{d_i}^\intercal \otimes \hat{\mathbf{s}}_{d_i}^\intercal\right)\mathbf{A} + \beta_i\beta_j\alpha\mathbf{s}_{d_i+1}^\intercal + \beta_i\beta_j\tilde{\mathbf{e}}_{d_i,3}^\intercal.
\end{aligned}$$

Now consider the value of $\tilde{\boldsymbol{\sigma}}_t$:

$$\begin{aligned}
\tilde{\boldsymbol{\sigma}}_t^\intercal &= (\boldsymbol{\sigma}_i^\intercal \otimes \hat{\boldsymbol{\sigma}}_j^\intercal)\mathbf{A} - \beta_j\boldsymbol{\sigma}_i^\intercal\mathbf{B}_{d_i,1} - \beta_i\hat{\boldsymbol{\sigma}}_j^\intercal\mathbf{B}_{d_i,2} + \beta_i\beta_j\mathbf{b}_{d_i,3}^\intercal \\
&= (\mathbf{k}_i^\intercal \otimes \hat{\mathbf{k}}_j^\intercal)\mathbf{A} + \beta_i\beta_j\alpha\mathbf{s}_{d_i+1}^\intercal + \left(\beta_i\beta_j\tilde{\mathbf{e}}_{d_i,3}^\intercal - \beta_j\boldsymbol{\sigma}_i^\intercal\mathbf{E}_{d_i,1} - \beta_i\hat{\boldsymbol{\sigma}}_j^\intercal\mathbf{E}_{d_i,2}\right) \in \mathbb{Z}_q^n.
\end{aligned}$$

By Eq. (7.1), a union bound, and the fact that $\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_j \in \mathbb{Z}_p^n$, we conclude that, with overwhelming probability,

$$\|\tilde{\mathbf{e}}_{d_i,3}\| \le \sqrt{\lambda}\tilde{\chi} \quad \text{and} \quad \|\boldsymbol{\sigma}_i^\intercal\mathbf{E}_{d_i,1}\|, \|\hat{\boldsymbol{\sigma}}_j^\intercal\mathbf{E}_{d_i,2}\| \le pn\sqrt{\lambda}\chi.$$

Since $\alpha > \lambda^{\omega(1)} \cdot p \cdot \max(\chi, \tilde{\chi})$ and $n = \mathrm{poly}(\lambda)$, by Lemma 3.1, over the choice of $\boldsymbol{\delta}_{t,1} \xleftarrow{R} \mathbb{Z}_q^n$, we now have

$$\begin{aligned}
\boldsymbol{\sigma}_t^\intercal &= \lfloor(\tilde{\boldsymbol{\sigma}}_t^\intercal + \boldsymbol{\delta}_{t,1}^\intercal)/\alpha\rceil \\
&= \lfloor\left((\mathbf{k}_i^\intercal \otimes \hat{\mathbf{k}}_j^\intercal)\mathbf{A} + \beta_i\beta_j\alpha\mathbf{s}_{d_i+1}^\intercal + \left(\beta_i\beta_j\tilde{\mathbf{e}}_{d_i,3}^\intercal - \beta_j\boldsymbol{\sigma}_i^\intercal\mathbf{E}_{d_i,1} - \beta_i\hat{\boldsymbol{\sigma}}_j^\intercal\mathbf{E}_{d_i,2}\right) + \boldsymbol{\delta}_{t,1}^\intercal\right)/\alpha\rceil \\
&= \lfloor\left((\mathbf{k}_i^\intercal \otimes \hat{\mathbf{k}}_j^\intercal)\mathbf{A} + \beta_i\beta_j\alpha\mathbf{s}_{d_i+1}^\intercal + \boldsymbol{\delta}_{t,1}^\intercal\right)/\alpha\rceil \\
&= \beta_i\beta_j\mathbf{s}_{d_i+1}^\intercal + \lfloor\left((\mathbf{k}_i^\intercal \otimes \hat{\mathbf{k}}_j^\intercal)\mathbf{A} + \boldsymbol{\delta}_{t,1}^\intercal\right)/\alpha\rceil \\
&= \beta_t\mathbf{s}_{d_t}^\intercal + \mathbf{k}_t^\intercal \in \mathbb{Z}_p^n.
\end{aligned}$$

As above, over the choice of $\boldsymbol{\delta}_{t,1} \xleftarrow{R} \mathbb{Z}_q^n$, the distribution of $\mathbf{k}_t$ is uniform over $\mathbb{Z}_p^n$. Since $\|\mathbf{s}_{d_t}\| \le \sqrt{\lambda}\chi$ with overwhelming probability, and $p > \lambda^{\omega(1)} \cdot \chi$, this relation holds over the integers with overwhelming probability and Eq. (7.4) holds.

Correctness now holds by induction over the gates of the circuit. □

Theorem 7.6 (Security). *Suppose $n = \mathrm{poly}(\lambda)$ and $p > \lambda^{\omega(1)} \cdot \chi$ and $\tilde{\chi} > \lambda^{\omega(1)} \cdot \chi^2$. Then, under the $\mathsf{LWE}_{n,n^2+n,q,\chi}$ assumption, Construction 7.4 is adaptively secure.*

Proof. We define the simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ as follows:

- $\mathcal{S}_1(1^\lambda, 1^d)$: On input the security parameter λ and the depth bound d, the simulator samples the following:

$$\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n^2 \times n}, \ \hat{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n \times n}, \ \mathbf{B}_{i,1}, \mathbf{B}_{i,2} \xleftarrow{R} \mathbb{Z}_q^{n \times n}, \ \mathbf{b}_{i,3} \xleftarrow{R} \mathbb{Z}_q^n, \ \mathbf{t}_i \xleftarrow{R} \mathbb{Z}_q^n, \ \hat{\boldsymbol{\sigma}}_i \xleftarrow{R} \mathbb{Z}_p^n$$

for all $i \in [0, d-1]$. Finally, it samples a PRF key $k \xleftarrow{R} \mathcal{K}$ and outputs

$$\mathsf{evk} = \left(k, \mathbf{A}, \hat{\mathbf{A}}, \left\{\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \mathbf{b}_{i,3}, \mathbf{t}_i, \hat{\boldsymbol{\sigma}}_i\right\}_{i \in [0,d-1]}\right).$$

and the state $\mathsf{st} = \perp$.

- $\mathcal{S}_2(\mathsf{st}, 1^\ell)$: On input the state st and the input length ℓ, the simulator outputs $\boldsymbol{\sigma} \xleftarrow{\mathrm{R}} (\mathbb{Z}_p^n)^\ell$.

Fix some efficient adversary $\mathcal{A}$ for the adaptive security game. To complete the proof, we define a sequence of hybrid experiments:

- Hyb_0: This is the adaptive security experiment with $b = 0$. Specifically, the experiment proceeds as follows:
  - On input the security parameter 1^λ, algorithm $\mathcal{A}$ outputs the depth bound 1^d and the global secret $\mathbf{s} \in \mathbb{Z}_p^n$. The challenger then proceeds as follows:
    - Sample a PRF key $k \xleftarrow{\mathrm{R}} \mathcal{K}$.
    - Sample the MAC key associated with input wires $\mathbf{s}_{\mathsf{in}} \leftarrow D_{\mathbb{Z},\chi}^n$.
    - For each $i \in [d-1]$, sample $\mathbf{s}_i, \hat{\mathbf{s}}_i \leftarrow D_{\mathbb{Z},\chi}^n$. Let $\mathbf{s}_0 = \mathbf{s}_{\mathsf{in}}$ and $\mathbf{s}_d = \mathbf{s}$.
    - Sample a matrix $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n^2 \times n}$, and for each $i \in [0, d-1]$, sample $\mathbf{E}_{i,1}, \mathbf{E}_{i,2} \leftarrow D_{\mathbb{Z},\chi}^{n \times n}$ and $\tilde{\mathbf{e}}_{i,3} \leftarrow D_{\mathbb{Z},\tilde{\chi}}^n$. Then compute the evaluation matrices
$$\begin{aligned}\mathbf{B}_{i,1} &= (\mathbf{I}_n \otimes \hat{\mathbf{s}}_i^\intercal)\mathbf{A} + \mathbf{E}_{i,1} \in \mathbb{Z}_q^{n \times n} \\ \mathbf{B}_{i,2} &= (\mathbf{s}_i^\intercal \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}_{i,2} \in \mathbb{Z}_q^{n \times n} \\ \mathbf{b}_{i,3}^\intercal &= (\mathbf{s}_i^\intercal \otimes \hat{\mathbf{s}}_i^\intercal)\mathbf{A} + \alpha \mathbf{s}_{i+1}^\intercal + \tilde{\mathbf{e}}_{i,3}^\intercal \in \mathbb{Z}_q^n.\end{aligned}$$
    - Next, sample $\hat{\mathbf{A}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$. Then, for each $i \in [0, d-1]$, sample $\hat{\mathbf{e}}_i \leftarrow D_{\mathbb{Z},\chi}^n$ and set $\mathbf{t}_i^\intercal = \mathbf{s}_i^\intercal \hat{\mathbf{A}} + \hat{\mathbf{e}}_i^\intercal - \alpha \hat{\mathbf{s}}_i^\intercal$.
    - For each $i \in [0, d-1]$, sample $\hat{\mathbf{k}}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p^n$ and compute $\hat{\boldsymbol{\sigma}}_i = \mathbf{s}_i + \hat{\mathbf{k}}_i \in \mathbb{Z}_p^n$.

    The challenger replies to $\mathcal{A}$ with the evaluation key evk
$$\mathsf{evk} = \left(k, \mathbf{A}, \hat{\mathbf{A}}, \left\{\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \mathbf{b}_{i,3}, \mathbf{t}_i, \hat{\boldsymbol{\sigma}}_i\right\}_{i \in [0,d-1]}\right).$$
  - Whenever algorithm $\mathcal{A}$ makes a query on a vector $\mathbf{x} \in \{0,1\}^\ell$, the challenger samples $\mathbf{k} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell}$ and responds with $\boldsymbol{\sigma} = \mathbf{x} \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}$.
  - At the end of the experiment, the adversary outputs a bit $b' \in \{0,1\}$, which is also the output of the experiment.
- Hyb_1: Same as Hyb_0 except when generating the evaluation key, the challenger samples $\hat{\boldsymbol{\sigma}}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p^n$ for all $i \in [0, d-1]$. Similarly, when responding to the adversary's queries, the challenger replies with $\boldsymbol{\sigma} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell}$.
- Hyb_2: Same as Hyb_1 except the challenger samples $\mathbf{B}_{i,1}, \mathbf{B}_{i,2} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$, $\mathbf{b}_{i,3} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ for all $i \in [0, d-1]$. This corresponds to the experiment with $b = 1$.

We write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output of an execution of Hyb_i with adversary $\mathcal{A}$. We now show that each adjacent pair of experiments is indistinguishable.

Lemma 7.7. *Hybrids* $\mathsf{Hyb}_0(\mathcal{A})$ *and* $\mathsf{Hyb}_1(\mathcal{A})$ *are identically distributed.*

Proof. These two experiments are identical. Namely, in Hyb_0, the challenger sets $\hat{\boldsymbol{\sigma}}_i = \mathbf{s}_i + \hat{\mathbf{k}}_i$ where $\hat{\mathbf{k}}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p^n$. Nothing else in the adversary's view depends on $\hat{\mathbf{k}}_i$, so the distribution of $\hat{\boldsymbol{\sigma}}_i$ in Hyb_0 is uniform over $\mathbb{Z}_p^n$, which matches the distribution in Hyb_1. Similarly, when responding to the adversary's queries, the challenger responds $\boldsymbol{\sigma} = \mathbf{x} \otimes \mathbf{s}_{\mathsf{in}} + \mathbf{k}$ where $\mathbf{k} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell}$. Since the challenger samples a fresh $\mathbf{k}$ on each query, the distribution of each $\boldsymbol{\sigma}$ is also uniform over $\mathbb{Z}_p^{n\ell}$, which is the distribution in Hyb_1. □

Lemma 7.8. *Suppose* $\tilde{\chi} > \lambda^{\omega(1)} \cdot \chi^2$. *Then, under the* $\mathsf{LWE}_{n,n^2+n,q,\chi}$ *assumption,* $\mathsf{Hyb}_1(\mathcal{A})$ *and* $\mathsf{Hyb}_2(\mathcal{A})$ *are computationally indistinguishable.*

Proof. We define an intermediate sequence of hybrid experiments:

- $\mathsf{Hyb}_{1,j,1}$ for all $j \in [0, d-1]$: Same as $\mathsf{Hyb}_{1,j-1,3}$, except when constructing the public parameters, the challenger samples $\mathbf{e}_{j,3} \leftarrow D^n_{\mathbb{Z},\chi}$ and then sets

$$\mathbf{b}^\intercal_{j,3} = \hat{\mathbf{s}}^\intercal_j \mathbf{B}_{j,2} + \mathbf{e}^\intercal_{j,3} + \alpha \mathbf{s}^\intercal_{j+1} + \tilde{\mathbf{e}}^\intercal_{j,3}.$$

- $\mathsf{Hyb}_{1,j,2}$ for all $j \in [0, d-1]$: Same as $\mathsf{Hyb}_{1,j,1}$, except the challenger now samples $\mathbf{B}_{j,2} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$ and $\mathbf{t}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$.
- $\mathsf{Hyb}_{1,j,3}$ for all $j \in [0, d-1]$: Same as $\mathsf{Hyb}_{1,j,2}$, except the challenger now samples $\mathbf{B}_{j,1} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$ and $\mathbf{b}_{j,3} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$.

We define $\mathsf{Hyb}_{1,-1,3} \equiv \mathsf{Hyb}_1$. By construction, we have $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_{1,d-1,3}$. It suffices to show that each pair of adjacent hybrids is indistinguishable:

- Consider $\mathsf{Hyb}_{1,j-1,3}$ and $\mathsf{Hyb}_{1,j,1}$. The only difference between these two distributions is the distribution of $\mathbf{b}^\intercal_{j,3}$. In $\mathsf{Hyb}_{1,j,1}$, the challenger sets

$$\begin{aligned}
\mathbf{b}^\intercal_{j,3} &= \hat{\mathbf{s}}^\intercal_j \mathbf{B}_{j,2} + \mathbf{e}^\intercal_{j,3} + \alpha \mathbf{s}^\intercal_{j+1} + \tilde{\mathbf{e}}^\intercal_{j,3} \\
&= \hat{\mathbf{s}}^\intercal_j ((\mathbf{s}^\intercal_j \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}_{j,2}) + \mathbf{e}^\intercal_{j,3} + \alpha \mathbf{s}^\intercal_{j+1} + \tilde{\mathbf{e}}^\intercal_{j,3} \\
&= (\mathbf{s}^\intercal_j \otimes \hat{\mathbf{s}}^\intercal_j)\mathbf{A} + \alpha \mathbf{s}^\intercal_{j+1} + (\hat{\mathbf{s}}^\intercal_j \mathbf{E}_{j,2} + \mathbf{e}^\intercal_{j,3} + \tilde{\mathbf{e}}^\intercal_{j,3}).
\end{aligned}$$

  Since $\hat{\mathbf{s}}_j \leftarrow D^n_{\mathbb{Z},\chi}$ and $\mathbf{E}_{j,2} \leftarrow D^{n \times n}_{\mathbb{Z},\chi}$, by Eq. (7.1), with overwhelming probability, $\|\hat{\mathbf{s}}^\intercal_j \mathbf{E}_{j,2}\| \le n\lambda\chi^2$. Similarly, with overwhelming probability, $\|\mathbf{e}_{j,3}\| \le \sqrt{\lambda}\chi$. Since the challenger samples $\tilde{\mathbf{e}}_{j,3} \leftarrow D_{\mathbb{Z},\tilde{\chi}}$ and $\tilde{\chi} \ge \lambda^{\omega(1)} \cdot \chi^2$, the statistical distance between the distribution of $\tilde{\mathbf{e}}_{j,3} \leftarrow D^n_{\mathbb{Z},\tilde{\chi}}$ and the distribution of $\hat{\mathbf{s}}^\intercal_j \mathbf{E}_{j,2} + \mathbf{e}^\intercal_{j,3} + \tilde{\mathbf{e}}^\intercal_{j,3}$ when $\tilde{\mathbf{e}}_{j,3} \leftarrow D^n_{\mathbb{Z},\tilde{\chi}}$ is negligible by Lemma 7.1. Hence, these two experiments are statistically indistinguishable.

- Suppose that $\mathsf{Hyb}_{1,j,1}(\mathcal{A})$ and $\mathsf{Hyb}_{1,j,2}(\mathcal{A})$ are computationally distinguishable. Then, we use $\mathcal{A}$ to construct a distinguisher $\mathcal{B}$ for the distributions in Corollary 7.3:
  - Algorithm $\mathcal{B}$ receives a challenge $(\mathbf{A}, \hat{\mathbf{A}}, \mathbf{Z}, \hat{\mathbf{z}})$.
  - Algorithm $\mathcal{B}$ starts running algorithm $\mathcal{A}$ on the security parameter 1^λ. Algorithm $\mathcal{A}$ outputs the depth bound 1^d and the global secret $\mathbf{s} \in \mathbb{Z}_p^n$.
  - Algorithm $\mathcal{B}$ samples $k \xleftarrow{\mathrm{R}} \mathcal{K}$.
  - For all $i < j$, algorithm $\mathcal{B}$ samples $\mathbf{B}_{i,1}, \mathbf{B}_{i,2} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times n}$ and $\mathbf{b}_{i,3}, \mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$.
  - For all $i > j$, algorithm $\mathcal{B}$ samples $\mathbf{s}_i, \hat{\mathbf{s}}_i \leftarrow D^n_{\mathbb{Z},\chi}$, $\mathbf{E}_{i,1}, \mathbf{E}_{i,2} \leftarrow D^{n \times n}_{\mathbb{Z},\chi}$, and $\tilde{\mathbf{e}}_{i,3} \leftarrow D^n_{\mathbb{Z},\tilde{\chi}}$. Then it sets

$$\begin{aligned}
\mathbf{B}_{i,1} &= (\mathbf{I}_n \otimes \hat{\mathbf{s}}^\intercal_i)\mathbf{A} + \mathbf{E}_{i,1} \\
\mathbf{B}_{i,2} &= (\mathbf{s}^\intercal_i \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}_{i,2} \\
\mathbf{b}^\intercal_{i,3} &= (\mathbf{s}^\intercal_i \otimes \hat{\mathbf{s}}^\intercal_i)\mathbf{A} + \alpha \mathbf{s}^\intercal_{i+1} + \tilde{\mathbf{e}}^\intercal_{i,3},
\end{aligned}$$

    where $\mathbf{s}_d = \mathbf{s}$. In addition, it samples $\hat{\mathbf{e}}_i \leftarrow D^n_{\mathbb{Z},\chi}$ and sets $\mathbf{t}^\intercal_i = \mathbf{s}^\intercal_i \hat{\mathbf{A}} + \hat{\mathbf{e}}^\intercal_i - \alpha \hat{\mathbf{s}}^\intercal_i$.
  - For $i = j$, algorithm $\mathcal{B}$ samples $\hat{\mathbf{s}}_j \leftarrow D^n_{\mathbb{Z},\chi}$, $\mathbf{E}_{j,1} \leftarrow D^{n \times n}_{\mathbb{Z},\chi}$, $\mathbf{e}_{j,3} \leftarrow D^n_{\mathbb{Z},\chi}$, and $\tilde{\mathbf{e}}_{j,3} \leftarrow D^n_{\mathbb{Z},\tilde{\chi}}$. Then, algorithm $\mathcal{B}$ sets

$$\begin{aligned}
\mathbf{B}_{j,1} &= (\mathbf{I}_n \otimes \hat{\mathbf{s}}^\intercal_j)\mathbf{A} + \mathbf{E}_{j,1} \\
\mathbf{B}_{j,2} &= \mathbf{Z} \\
\mathbf{b}^\intercal_{j,3} &= \hat{\mathbf{s}}^\intercal_j \mathbf{B}_{j,2} + \alpha \mathbf{s}^\intercal_{j+1} + \mathbf{e}^\intercal_{j,3} + \tilde{\mathbf{e}}^\intercal_{j,3}.
\end{aligned}$$

    It also sets $\mathbf{t}_j = \hat{\mathbf{z}} - \alpha \hat{\mathbf{s}}_j$.
  - For all $i \in [0, d-1]$, algorithm $\mathcal{B}$ samples $\hat{\boldsymbol{\sigma}}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p^n$.

– Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with the evaluation key evk

$$\mathsf{evk} = \left(k, \mathbf{A}, \hat{\mathbf{A}}, \left\{\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \mathbf{b}_{i,3}, \mathbf{t}_i, \hat{\boldsymbol{\sigma}}_i\right\}_{i\in[0,d-1]}\right).$$

– Whenever algorithm $\mathcal{A}$ makes an evaluation query on an input $\mathbf{x} \in \{0,1\}^\ell$, algorithm $\mathcal{B}$ responds with $\boldsymbol{\sigma} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell}$.

– At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ which $\mathcal{B}$ also outputs.

It suffices to analyze the distribution of $\mathbf{B}_{j,2}$ and $\mathbf{t}_j$. All other components are distributed exactly as in $\mathsf{Hyb}_{1,j,1}$ and $\mathsf{Hyb}_{1,j,2}$. We consider the two possibilities:

– Suppose $\mathbf{Z} = (\mathbf{s}_j^\top \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}$ and $\hat{\mathbf{z}} = \mathbf{s}_j^\top \hat{\mathbf{A}} + \hat{\mathbf{e}}^\top$ where $\mathbf{s}_j \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\chi}^{n\times n}$, and $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\chi}^n$. This corresponds to the distribution of $\mathsf{Hyb}_{1,j,1}$ and algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_{1,j,1}(\mathcal{A}) = 1]$.

– Suppose $\mathbf{Z} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times n}$ and $\hat{\mathbf{z}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$. This corresponds to the distribution $\mathsf{Hyb}_{1,j,2}$ and algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_{1,j,2}(\mathcal{A}) = 1]$.

We conclude that algorithm $\mathcal{B}$ distinguishes the distributions in Corollary 7.3 with the same advantage as $\mathcal{A}$ and the claim follows.

- Suppose that $\mathsf{Hyb}_{1,j,2}(\mathcal{A})$ and $\mathsf{Hyb}_{1,j,3}(\mathcal{A})$ are computationally distinguishable. Then, we use $\mathcal{A}$ to construct a distinguisher $\mathcal{B}$ for the distributions in Lemma 7.2:

  – Algorithm $\mathcal{B}$ receives a challenge $(\mathbf{A}, \mathbf{B}_{j,2}, \mathbf{Z}, \hat{\mathbf{z}})$.

  – Algorithm $\mathcal{B}$ starts running algorithm $\mathcal{A}$ on the security parameter 1^λ. Algorithm $\mathcal{A}$ outputs the depth bound 1^d and the global secret $\mathbf{s} \in \mathbb{Z}_p^n$.

  – Algorithm $\mathcal{B}$ samples $k \xleftarrow{\mathrm{R}} \mathcal{K}$.

  – For all $i < j$, algorithm $\mathcal{B}$ samples $\mathbf{B}_{i,1}, \mathbf{B}_{i,2} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times n}$ and $\mathbf{b}_{i,3}, \mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$.

  – For all $i > j$, algorithm $\mathcal{B}$ samples $\mathbf{s}_i, \hat{\mathbf{s}}_i \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E}_{i,1}, \mathbf{E}_{i,2} \leftarrow D_{\mathbb{Z},\chi}^{n\times n}$, and $\tilde{\mathbf{e}}_{i,3} \leftarrow D_{\mathbb{Z},\tilde{\chi}}^n$. Then it sets

$$\begin{aligned}
\mathbf{B}_{i,1} &= (\mathbf{I}_n \otimes \hat{\mathbf{s}}_i^\top)\mathbf{A} + \mathbf{E}_{i,1} \\
\mathbf{B}_{i,2} &= (\mathbf{s}_i^\top \otimes \mathbf{I}_n)\mathbf{A} + \mathbf{E}_{i,2} \\
\mathbf{b}_{i,3}^\top &= (\mathbf{s}_i^\top \otimes \hat{\mathbf{s}}_i^\top)\mathbf{A} + \alpha\mathbf{s}_{i+1}^\top + \tilde{\mathbf{e}}_{i,3}^\top,
\end{aligned}$$

    where $\mathbf{s}_d = \mathbf{s}$. In addition, it samples $\hat{\mathbf{e}}_i \leftarrow D_{\mathbb{Z},\chi}^n$ and sets $\mathbf{t}_i^\top = \mathbf{s}_i^\top \hat{\mathbf{A}} + \hat{\mathbf{e}}_i^\top - \alpha\hat{\mathbf{s}}_i^\top$.

  – For $i = j$, algorithm $\mathcal{B}$ sets $\mathbf{B}_{j,1} = \mathbf{Z}$ and $\mathbf{b}_{j,3}^\top = \hat{\mathbf{z}}^\top + \alpha\mathbf{s}_{j+1}^\top + \tilde{\mathbf{e}}_{j,3}^\top$ where $\tilde{\mathbf{e}}_{j,3} \leftarrow D_{\mathbb{Z},\tilde{\chi}}^n$. Recall that $\mathbf{B}_{j,2}$ is set from the challenge. It then samples $\mathbf{t}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$.

  – For all $i \in [0, d-1]$, algorithm $\mathcal{B}$ samples $\hat{\boldsymbol{\sigma}}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p^n$.

  – Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with the evaluation key evk

$$\mathsf{evk} = \left(k, \mathbf{A}, \hat{\mathbf{A}}, \left\{\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \mathbf{b}_{i,3}, \mathbf{t}_i, \hat{\boldsymbol{\sigma}}_i\right\}_{i\in[0,d-1]}\right).$$

  – Whenever algorithm $\mathcal{A}$ makes an evaluation query on an input $\mathbf{x} \in \{0,1\}^\ell$, algorithm $\mathcal{B}$ responds with $\boldsymbol{\sigma} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n\ell}$.

  – At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ which $\mathcal{B}$ also outputs.

It suffices to analyze the distribution of $\mathbf{B}_{j,1}$ and $\mathbf{b}_{j,3}$. All other components are distributed exactly as in $\mathsf{Hyb}_{1,j,2}$ and $\mathsf{Hyb}_{1,j,3}$. We consider the two possibilities:

– Suppose $\mathbf{Z} = (\mathbf{I}_n \otimes \mathbf{s}_j^\top)\mathbf{A} + \mathbf{E}$ and $\hat{\mathbf{z}} = \mathbf{s}_j^\top \mathbf{B}_{j,2} + \hat{\mathbf{e}}^\top$ where $\mathbf{s}_j \leftarrow D_{\mathbb{Z},\chi}^n$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\chi}^{n\times n}$, and $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\chi}^n$. This corresponds to the distribution of $\mathsf{Hyb}_{1,j,2}$ and algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_{1,j,2}(\mathcal{A}) = 1]$.

– Suppose $\mathbf{Z} \xleftarrow{R} \mathbb{Z}_q^{n \times n}$ and $\hat{\mathbf{z}} \xleftarrow{R} \mathbb{Z}_q^n$. This corresponds to the distribution $\mathsf{Hyb}_{1,j,3}$ and algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_{1,j,3}(\mathcal{A}) = 1]$.

We conclude that algorithm $\mathcal{B}$ distinguishes the distributions in Lemma 7.2 with the same advantage as $\mathcal{A}$ and the claim follows.

Lemma 7.8 now follows by a standard hybrid argument. □

Adaptive security now follows by combining Lemmas 7.7 and 7.8. □

Acknowledgements

We thank Yuval Ishai for many helpful comments and discussions. Akshayaram Srinivasan and Siddharth Agarwal are supported in part by the NSERC Discovery Grant RGPIN-2024-03928. Akshayaram Srinivasan is additionally supported by a SDF academic research award. David J. Wu is supported by NSF CNS-2140975, CNS-2318701, a Sloan Fellowship, a Microsoft Research Faculty Fellowship, a Google Research Scholar Award, an Amazon Research Award, and a gift from the Stellar Development Foundation. Abhishek Jain is supported in part by NSF CAREER 1942789, Johns Hopkins University Catalyst award, JP Morgan Faculty Award, and research gifts from the Ethereum Foundation, Stellar Development Foundation, and Cisco. Part of this work was conducted while Abhishek Jain and David J. Wu were visiting the Simons Institute for the Theory of Computing and supported in part by a grant from the UC Noyce Initiative.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, 2009.

[AJJM20] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fully-homomorphic encryption in the plain model. In *TCC*, 2020.

[AJJM21] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Unbounded multi-party computation from learning with errors. In *EUROCRYPT*, 2021.

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Succinct oblivious tensor evaluation and applications: Adaptively-secure laconic function evaluation and trapdoor hashing for all circuits. In *STOC*, 2025.

[ARS24] Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In *EUROCRYPT*, 2024.

[BCD+25] Pedro Branco, Arka Rai Choudhuri, Nico Döttling, Abhishek Jain, Giulio Malavolta, and Akshayaram Srinivasan. Black-box non-interactive zero knowledge from vector trapdoor hash. In *EUROCRYPT*, 2025.

[BDE+18] Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. In *ASIACRYPT*, 2018.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC*, 2019.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, 2014.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *ACM CCS*, 2016.

[BGMM20] James Bartusek, Sanjam Garg, Daniel Masny, and Pratyay Mukherjee. Reusable two-round MPC from DDH. In *TCC*, 2020.

[BGSZ22] James Bartusek, Sanjam Garg, Akshayaram Srinivasan, and Yinuo Zhang. Reusable two-round MPC from LPN. In *PKC*, 2022.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[BJKL21] Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In *EUROCRYPT*, 2021.

[BJSS25] Elette Boyle, Abhishek Jain, Sacha Servan-Schreiber, and Akshayaram Srinivasan. Simultaneous-message and succinct secure computation. In *EUROCRYPT*, 2025.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In *CRYPTO*, 2020.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT*, 2019.

[BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: multiparty reusable non-interactive secure computation. In *TCC*, 2020.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, 2012.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In *TCC*, 2017.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In *CRYPTO*, 2017.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4), 2004.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, 1995.

[CHP25] Geoffroy Couteau, Aditya Hegde, and Sihang Pu. Enhanced trapdoor hashing from DDH and DCR. In *EUROCRYPT*, 2025.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, 2001.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO*, 2019.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO*, 2016.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *STOC*, 1994.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *STOC*, 1987.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *FOCS*, 2023.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *CRYPTO*, 2011.

[ILL25a] Yuval Ishai, Hanjun Li, and Huijia Lin. Succinct homomorphic MACs from groups and applications. In *FOCS*, 2025.

[ILL25b] Yuval Ishai, Hanjun Li, and Huijia Lin. A unified framework for succinct garbling from homomorphic secret sharing. In *CRYPTO*, 2025.

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In *EUROCRYPT*, 2021.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *FOCS*, 1997.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS*, 2013.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, 2016.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, 1990.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In *FOCS*, 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, 1999.