Simultaneous-Message and Succinct Secure Computation

Elette Boyle[1,2], Abhishek Jain[1,3], Sacha Servan-Schreiber[4], and Akshayaram Srinivasan[5]

[1] NTT Research
[2] Reichman University
[3] JHU
[4] MIT
[5] University of Toronto

Abstract. We put forth and instantiate a new primitive we call *simultaneous-message and succinct (SMS)* secure computation. An SMS scheme enables a *minimal* communication pattern for secure computation in the following scenario: Alice has a large private input X, Bob has a small private input y, and Charlie wants to learn $f(X, y)$ for some public function f.

Given a common reference string (CRS) setup phase, an SMS scheme for a function f is instantiated with two parties holding inputs X and y, and has the following structure:

- The parties *simultaneously* exchange a single message.
- Communication is *succinct*, scaling sublinearly in the size of X and the output $f(X, y)$.
- Without further interaction, the parties can locally derive additive secret shares of $f(X, y)$.

In this way, an SMS scheme incurs a communication cost that is only twice that of the function output length. Importantly, the size of Alice's message does not grow with the size of her input X, and both Alice's and Bob's first-round messages grow sublinearly in the size of the output. Additionally, Alice's or Bob's view provides no information about the other party's input besides the output of $f(X, y)$, even if colluding with Charlie.

We obtain the following results:

- Assuming Learning With Errors (LWE), we build an SMS scheme supporting evaluation of depth-d circuits, where Alice's message is of size $|f(X, y)|^{2/3} \cdot \mathsf{poly}(\lambda, d)$, Bob's message is of size $(|y| + |f(X, y)|^{2/3}) \cdot \mathsf{poly}(\lambda, d)$, and λ is the security parameter. We can further extend this to support all functions by assuming the circular security of LWE.
- Assuming sub-exponentially secure indistinguishability obfuscation, in conjunction with other standard assumptions, we build an SMS scheme supporting arbitrary polynomial-sized *batch* functions of the form $(f(x_1, y), \ldots, f(x_L, y))$, for $X = (x_1, \ldots, x_L)$. The size of Alice's and Bob's messages in this construction is $\mathsf{poly}(\lambda)$ and $\mathsf{poly}(\lambda, |f|, \log L)$, respectively.

We show that SMS schemes have several immediate applications. An SMS scheme gives:

1. A direct construction of trapdoor hash functions (TDH) (Döttling et al., Crypto'19) for the same class of functions as the one supported by the SMS scheme.
2. A simple and generic compiler for obtaining compact, rate-1 fully homomorphic encryption (FHE) from any non-compact FHE scheme.
3. A simple and generic compiler for obtaining correlation-intractable (CI) hash functions that are secure against all efficiently-searchable relations.

In turn, under the LWE assumption, we obtain the first construction of TDH for all functions and generic approaches for obtaining rate-1 FHE and CI hashing. We also show that our $i\mathcal{O}$-based construction gives an alternative approach for two-round secure computation with communication succinctness in the output length (Hubáček and Wichs, ITCS'15).

Table of Contents

1 Introduction

Consider the following scenario: Alice has a *large* private input X, Bob has a *small* private input y, and Charlie wants to learn the output $f(X, y)$ of some public function f evaluated over the inputs of Alice and Bob. To achieve this with optimal communication cost, Bob can simply send his input to Alice, who then computes the output $f(X, y)$, and sends it to Charlie. This simple, but clearly insecure, protocol achieves the following communication complexity:

- The communication between Alice and Bob is only $|y|$, and in particular, independent of the length of Alice's input X and the output of the function $f(X, y)$.
- The total communication to Charlie is simply the length of the function output (and, in particular, independent of Alice and Bob's input lengths).

Furthermore, this protocol requires only a single message from Bob to Alice and then from Alice to Charlie. In this paper, we ask the following.

> *Can we design a secure computation protocol that preserves, to the extent possible, the communication complexity and the communication pattern of the above insecure protocol?*

Specifically, we will consider secure computation protocols [Yao86, GMW87] with security against semi-honest adversaries who may corrupt either of the two input parties (Alice or Bob) together with the output party (Charlie).

Simultaneous-Message and Succinct Secure Computation. To answer this question, we investigate a minimal model of computation that we refer to as *simultaneous-message and succinct* (SMS) secure computation. In an SMS scheme, following a setup phase that outputs a common reference string (CRS),[6] the interaction proceeds as follows:

- **Encode:** Alice and Bob encode their private inputs into public encoding pe_A and pe_B, respectively, and exchange these encodings in a simultaneous round of communication.
- **Decode:** Given her private state and public encodings, Alice (resp., Bob) computes a share z_A (resp., z_B) and sends it to Charlie, who can locally reconstruct the output.

Communication between parties in both encode and decode phases is simultaneous. Moreover, for any big input X, small input y, and a function f, we require that the decoded values z_A and z_B computed by Alice and Bob form an *additive sharing* of $f(X, y)$, which allows Charlie to reconstruct $f(X, y) = z_A \oplus z_B$. Note that because additive shares are information-theoretically the same size as the output, the communication cost of the decode phase is only twice that of the insecure protocol. We further require the following *succinctness* property for the encode phase: the size of the public encodings pe_A and pe_B is succinct with respect to Alice's input length $|X|$ and the function output length.[7] Finally, we require standard simulation-based security against semi-honest adversaries.

In summary, compared to the "optimal" insecure protocol, SMS requires two additional messages: one from Alice to Bob, and another message from Bob to Charlie. In the secure setting, these additional messages are necessary to prevent input-resetting attacks [HLP11]. However, we show that Bob's message can, in some cases, be as short as the security parameter λ. In this sense, the communication model of SMS is *minimal*.

We investigate the feasibility of constructing SMS schemes and present positive results as well as several applications. Before we proceed to describe our results, we first compare SMS with some related notions in cryptography.

Comparison with succinct protocols. SMS can be viewed as extending the notion of private simultaneous messages [FKN94] along two dimensions. First, SMS allows *collusion* between an input party and the output party (and hence, requires an additional round of communication). Second, SMS requires succinct communication in the input size and function description.

The study of "circuit-succinct" secure communication has a rich history in cryptography. Arguably, it was most popularized by fully homomorphic encryption [Gen09], which yields secure communication with communication independent of the size of the circuit representation of the function. If we relax the *simultaneous-message* requirement, and allow one party to send its message after the other, we

[6] In our constructions, we only need a common random string.

[7] Note that by communication complexity lower bounds, we cannot expect to achieve communication sublinear in the input lengths of both parties.

can also obtain "input-succinctness" in the size of one of the party's inputs by using FHE or its "dual" notion of laconic function evaluation [CDG+17, QWW18]. However, neither of these two approaches yield SMS. For example, using (circuit-private) FHE, one can attempt to design an SMS scheme as follows: (1) Bob sends an encryption of his input y to Alice, (2) Alice homomorphically computes the encryption of $f(X, y)$ and sends it to Charlie, and (3) Bob sends his secret key to Charlie. Clearly, this protocol is insecure against collusion with Alice, since Charlie gets the secret key. Furthermore, the output phase requires not admit additive reconstruction and requires communicating more than just the function output.

SMS bears resemblance to the notion of homomorphic secret sharing (HSS) [BGI16], most notably in the requirement of additive reconstruction and succinctness in the circuit size. However, SMS and HSS are incomparable. For one, HSS (and the more powerful notion of spooky encryption [DHRW16]) does not require succinctness in the input length. Moreover, HSS and spooky encryption support an *adaptive* choice of functions: the function to be computed can be determined *after* the input encoding phase. SMS, in contrast, does not necessarily require this property (although, as we will discuss later, one of our constructions achieves it).

Because of the input-succinctness requirement, SMS is closer to the recently-proposed notion of *succinct* HSS [ARS24], which extends HSS to require succinctness with respect to one of the inputs (in addition to succinctness with respect to the function description). However, succinct HSS still requires a correlated randomness setup (thus, it cannot support the minimal communication pattern of SMS) and current schemes are only suitable for very restricted function classes.

Comparison with two-round secure computation. We note that SMS is stronger than two-round secure computation [Yao86, BL18, GS18] because of the input and circuit succinctness properties in addition to the additive reconstruction properties. Indeed, because of the additive reconstruction requirement alone, SMS—even for a single AND computation—implies non-interactive key exchange via the reduction of Boyle et al. [BGI+18], and therefore is black-box separable from oblivious transfer [GKM+00].

Applications. SMS implies the previously studied notion of trapdoor hashing (TDH) Döttling et al. [DGI+19] [DGI+19]. A trapdoor hash scheme is a protocol between two parties—a sender and a receiver—in the common reference string (CRS) model. Given the CRS (referred to as the hash key in the original work), the sender can compute a digest d of its input X, while the receiver can encode a private function f into an evaluation key ek. Given these encoded values, the sender and the receiver can compute an additive secret sharing of $f(X, y)$. A TDH scheme requires two properties: (1) sender succinctness, namely, the size of the digest d must be sublinear in the sender's input length, and (2) receiver privacy, namely, the evaluation key must hide the function f.

It is easy to see that SMS implies TDH by assigning the role of the sender to Alice and the role of the receiver to Bob. In fact, SMS is *stronger* than TDH since the function f is "decoupled" from Bob's input, and hence, we can require both Alice and Bob's messages to be of size sublinear in Alice's input length. Döttling et al. [DGI+19] (and subsequent works [BKM20, GHO20]) constructed TDH schemes for linear functions from a variety of standard assumptions, and constructing TDH for larger function classes has remained open. As we will discuss shortly, our positive results on SMS (combined with the above implication) break this barrier by realizing the first TDH for all depth-d circuits from the learning with errors assumption [Reg05]. By additionally assuming and the circular security of LWE, we get the first TDH for all polynomial-size circuits.

We also demonstrate direct applications of SMS to other powerful primitives, including rate-1 fully homomorphic encryption [BDGM19, GH19], correlation-intractable hash functions, and output-succinct secure computation.

1.1 Our results

We initiate the study of SMS protocols and present several positive results and applications.

SMS from LWE. Our first result is an SMS scheme for all depth-bounded computations, assuming the hardness of learning with errors (LWE). Specifically, for all depth-d circuits, we construct an SMS scheme where the communication complexity of the encode phase grows with d, but is otherwise sublinear in the input and output length of the function being computed. This first result is captured in the following theorem:

	Assumption	Input Succinctness	Output Succinctness	Function Class
Spooky Encryption [DHRW16]	LWE / $i\mathcal{O}$+DDH	✗	✓	All Circuits
NIVOLE [OSY21, CDD+24]	DCR / LWE	✗	✗	Degree-2
Succinct NIVOLE [ARS24, BCM+24]	DCR / QR / LWE	✓	✓	Degree-2
Section 5	LWE	✓	✓	Depth-d Circuits
Section 6	$i\mathcal{O}$+SSB	✓	✓	All Batch Circuits
Section 7	Circular LWE	✓	✓	All Circuits

Table 1. Constructions of SMS.

Theorem 1 (Informal)**.** *Let $\mathcal{F}$ be the family of all functions that are computable by depth-d circuits. Assuming the hardness of learning with errors (with a superpolynomial modulus-to-noise ratio), there exists an SMS scheme for any function $f \in \mathcal{F}$, where in the encode phase, the size of Alice's message is $|f(X,y)|^\epsilon \cdot \mathsf{poly}(\lambda, d)$ and the size of Bob's message is $(|y| + |f(X,y)|^\epsilon) \cdot \mathsf{poly}(\lambda, d)$. Here, λ is the security parameter and $\epsilon = (2/3)$. By additionally assuming the circular-security of LWE, we obtain an SMS scheme where the message size is independent of the circuit size.*

SMS from Indistinguishability Obfuscation. Our second result is an SMS scheme for all *batch* computations, assuming the existence of sub-exponentially-secure indistinguishability obfuscation [BGI+01, GGH+13, JLS21], sub-exponentially secure one-way functions and somewhere statistically-binding (SSB) hash functions [HW15].[8] In the batch setting, Alice holds as input a long vector $X := (x_1, \ldots, x_L)$ and Bob holds an input y; and they wish to compute $f(x_1, y), \ldots, f(x_L, y)$ given a public function f in some function family $\mathcal{F}$. Here, we relax the succinctness requirement for the encoding phase: The total communication must be at most polylogarithmically dependent on the batch size L, but can grow with the size of the circuit description.

Theorem 2 (Informal)**.** *Let $\mathcal{F}$ be the family of all functions that are computable by polynomial-size circuits. Assuming the existence of (1) sub-exponentially secure indistinguishability obfuscation, (2) sub-exponentially secure one-way functions, (3) somewhere statistically-binding hash functions (with perfect binding), and (4) the existence of injective one-way functions, there exists an SMS scheme that supports batch computation of functions in $\mathcal{F}$. For any batch size L, the size of both Alice' and Bob's messages in the encode phase is $\mathsf{poly}(\lambda, |f|, \log L)$, where λ is the security parameter.*

The above protocol supports *adaptive* choice of functions during the decode phase. That is, the parties can compute their public encodings in the encode phase *independently* of the function. Then, the public encodings can be *reused* for computing the decode phase for any choice of batch functions.

We next discuss applications of the above results.

Application I: Trapdoor hashing beyond linear functions. Assuming the hardness of LWE, we obtain a construction of trapdoor hash functions for all depth-d circuits. By additionally assuming the circular-security of LWE, we obtain a construction of trapdoor hash functions for all polynomial-size circuits. This significantly improves upon the state of the art, where trapdoor hash functions were only known for linear functions. This result follows immediately from Theorem 1 combined with the aforementioned direct implication from SMS to trapdoor hashing, so we omit a formal construction in this paper.

Application II: Rate-1 fully homomorphic encryption, generically. In a rate-1 fully homomorphic encryption scheme, the message to ciphertext length ratio (i.e., rate) is $1 - o(1)$. A rate-1 FHE scheme was first constructed under the LWE assumption by Brakerski, Döttling, Garg, and Malavolta [BDGM19] and Gentry and Halevi [GH19]. At a high level, their constructions intricately combine FHE schemes with rate-1 linearly homomorphic encryption to compress ciphertexts.

We show that an SMS scheme can be used to transform *any* FHE scheme into a rate-1 FHE scheme. Our construction is quite simple and generic. We briefly sketch the transformation below and provide more details in Section 8.

[8] We additionally require the existence of injective one-way functions and perfect binding for the SSB hash.

Consider any FHE scheme with a poor rate and let the decryption algorithm of this FHE scheme be described by a function f that takes as input a ciphertext ct and a secret key sk, and outputs the message. To compress the ciphertexts of this scheme, we use an SMS scheme that performs the computation of f. In more detail:

- *New keys.* Let $(\mathsf{pk}, \mathsf{sk})$ be a public key and secret key pair of the underlying FHE scheme. The public key of the new FHE scheme consists of the tuple $(\mathsf{pk}, \mathsf{pe}_B)$ where pe_B is Bob's public encoding output by the SMS scheme and computed using the secret key sk as his private input. The new secret key is simply Bob's private state st_B output by the SMS scheme.
- *Ciphertext compression.* Let ct be a homomorphically-evaluated ciphertext ct of the underlying FHE scheme with poor rate. To compress this ciphertext, we first use ct as the input to the SMS scheme to compute Alice's public encoding pe_A in the encode phase. Then, we use Bob's public encoding pe_B (which is now part of the new public key) to compute Alice's decoded value z_A. The compressed ciphertext is simply the tuple (pe_A, z_A).
- *Decryption.* To decrypt the ciphertext (pe_A, z_A), we first use pe_A and Bob's private state st_B (which is part of the new secret key) to compute Bob's decoded value z_B. The plaintexts are recovered as $z_A \oplus z_B$.

Arguing correctness and security. The correctness of the scheme follows by inspection. Intuitively, we use SMS to decrypt the ciphertexts, which results in additive shares of the messages. Because the public encodings are sublinear in the size of the ciphertext, the rate asymptotically approaches 1. The security of the scheme follows from the security of the underlying FHE scheme as well as security for Bob in the SMS scheme (we do not require security for Alice here). The full transformation and proof of security are provided in Section 8.

Application III: Correlation-Intractable Hash Functions, generically. Correlation-intractable (CI) hash functions [CGH98] are functions whose input-output pairs behave in a similar way to a random function in that they do not satisfy any "bad" correlations. Specifically, a hash function family H_{hk} is said to be correlation intractable for a relation class $\mathcal{R}$, if for any relation $R \in \mathcal{R}$, no efficient adversary given the hash key hk can find an input-output pair that satisfies R.

Recently, CI hashing has found numerous applications in cryptography, most notably in achieving new constructions of non-interactive zero knowledge proofs (see, e.g., [CCH+19, PS19, JJ21]) and succinct non-interactive arguments (see, e.g., [CHK+19, CJJ21, JKKZ21, CJJ22]) in the standard model. These applications are obtained by using CI hashing to securely instantiate the Fiat–Shamir paradigm [FS87] for round-collapsing interactive proofs.

In Section 9, we show a simple and generic construction of CI hashing for efficiently-searchable relations from SMS. Using Theorem 1, we obtain CI hashing for relations searchable by depth-bounded circuits from LWE. Previously, CI hashing from LWE was known using the work of Peikert and Shiehian [PS19]. However, our construction of CI hashing from SMS is generic, and uses the observation of Brakerski, Koppula, and Mour [BKM20] that CI hashing can be constructed from trapdoor hashing. See Section 9 for details on our transformation.

1.2 Related work

In this section, we discuss some connections between SMS and other related notions in cryptography.

Output-succinct secure computation. The work of Hubáček and Wichs [HW15] investigated the feasibility of secure computation with *total* communication complexity that is sublinear in the function output length. Using indistinguishability obfuscation ($i\mathcal{O}$) [BGI+01, GGH+13] and other standard assumptions, they constructed interactive protocols with sublinear communication using a large common random string (or large random tapes for the parties) of length proportional to the output length. In fact, they demonstrated that the use of program obfuscation (with a large CRS) is necessary for this task. This implication does *not* hold for SMS since the communication between the input parties and the output party (Charlie) does grow with the output length. In particular, SMS only requires the encodings (first round messages) to be succinct in the input and output size. Nonetheless, in Section 7, we show that our $i\mathcal{O}$-based construction of SMS can be extended to have a

succinct second-round message from Bob to Charlie. This extension gives an alternative construction of the secure computation protocols considered by Hubáček and Wichs.

Succinct homomorphic secret sharing. The recent work of Abram, Roy, and Scholl [ARS24] constructs *succinct* homomorphic secret sharing (HSS). Similarly to the PSM model, HSS allows a correlated setup to take place between between Alice and Bob. However, the additional succinctness requirement considered in [ARS24], in conjunction with the collusion guarantees of HSS, make the notion more closely related to SMS. In the construction of succinct HSS, Alice with a large vector $\mathbf{a}$ and a short input x, and Bob with short input y, can compute a function of the form $\langle \mathbf{a}, f(x,y)\rangle$ with communication that is sublinear in $|\mathbf{a}|$. Concretely, their constructions result in $O_\lambda(\sqrt{|\mathbf{a}|} + |x| + |y|)$ communication and require three rounds of interaction when including the correlated setup between the parties.

Indeed, the core primitive used to realize succinct HSS, that turns out to also be instrumental in realizing SMS (cf. Section 5.2), is the recently introduced notion of succinct non-interactive VOLE (NIVOLE) [ARS24, BCM+24]. We show that succinct NIVOLE can be cast as an SMS scheme for degree-2 functions, given that it does not require a correlated setup between parties [BCM+24].

We give a comparison of our results with related notions in Table 1.

1.3 Paper organization

We begin with a technical overview of our constructions in Section 2. Section 3 introduces the necessary preliminaries and notation. In Section 4, we formally define SMS and explore its relationships with other primitives. Section 5 presents our LWE-based construction for depth-d circuits, which we extend to all circuits in Section 7. We then detail our $i\mathcal{O}$-based construction for batch function evaluations in Section 6. Section 7 covers various optimizations and extensions, including an extension to our $i\mathcal{O}$-based construction of SMS that compresses the second-round message to achieve output-succinct secure computation. Finally, we demonstrate applications: a compiler to rate-1 FHE in Section 8 and a compiler to correlation-intractable hashing in Section 9.

2 Technical Overview

In this section, we give an overview of our constructions of SMS schemes from LWE (Section 2.1) and from indistinguishability obfuscation (Section 2.2).

2.1 Construction from LWE

The high-level idea is to start with the ABE scheme of Boneh et al. [BGG+14], which has two useful algorithms EvalPK and EvalCT that can be used in a "black-box" way [GVW15, QWW18]. The CRS consists of α matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\alpha$. Let C be a depth-d arithmetic circuit producing m-bit outputs and let C_i be the circuit that outputs the i-th bit of C. The two algorithms have the following syntax:

- $\mathsf{EvalPK}(\mathsf{crs}, C) \to \mathbf{A}_C$. Takes as input the CRS and a circuit description C producing m bit outputs; it outputs a list $\mathbf{A}_C := (\mathbf{A}_{C_i})_{i=1}^m$ of size $m \cdot \mathsf{poly}(\lambda, d)$.
- $\mathsf{EvalCT}(\mathsf{crs}, \mathsf{ct}, C, x) \to \mathbf{w}_C$. Takes as input a ciphertext vector $\mathsf{ct} := (\mathbf{s}^\top(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^\top)_{i\in[\alpha]}$ encrypting each bit of the input x to the circuit C; it outputs $\mathbf{w}_C := (\mathbf{s}^\top(\mathbf{A}_{C_i} + C_i(x) \cdot \mathbf{G}) + \tilde{\mathbf{e}}_i^\top)_{i\in[m]}$, encrypting each bit of the output $C(x)$.

Here, $\mathbf{G}$ is the standard gadget matrix (see Definition 12 for a formal definition).

Inspired by the ideas that underpin the predicate encryption scheme of Gorbunov, Vaikuntanathan, and Wee [GVW15] and laconic function evaluation of Quach, Wee, and Wichs [QWW18], our idea is to use EvalPK to commit Alice to her large input X by hardcoding it into a large circuit C. More concretely, Alice defines C to be the circuit that only takes as input an FHE-encrypted input of y and outputs the (encrypted) FHE evaluation of $f(X, y)$.[9] Alice sends Bob $\mathbf{A}_C$—the output of EvalPK when evaluated on her large circuit C. Bob, on the other hand, encrypts his input y under a suitable FHE scheme and sends it to Alice. Surprisingly, we will show that with just a few tweaks, these values are sufficient to construct an SMS scheme.

[9] We need to assume that the FHE evaluation of a depth-d circuit can itself be represented by a depth-d circuit, which is the case for existing LWE-based FHE schemes.

We now proceed to explain our initial attempt to realize SMS. While this first approach does not work out-of-the-box, it provides us with the right insights and framework to build off of.

Setup. Let α denote the length (in bits) of an FHE ciphertext encrypting an ℓ-bit input. For now, we define the common *random* string crs to simply consist of α random matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_\alpha)$; later, we will add more matrices to crs, while still keeping it uniformly random.

Step 1: Generating the encodings. Alice computes $\mathsf{EvalPK}(\mathsf{crs}, C)$ to obtain $\mathbf{A}_C$, where C is as defined above. This $\mathbf{A}_C$ implicitly commits Alice to X. Bob samples a secret vector $\mathbf{s}$ such that the first coordinate of $\mathbf{s}$ is 1, and generates a secret key sk for a leveled fully homomorphic encryption scheme (cf. Definition 2). He encrypts his input y under the FHE scheme using sk to obtain the bits of the ciphertext $\mathsf{ct}_{\mathsf{FHE}} := (\mathsf{ct}_{\mathsf{FHE}}^{(1)} \| \ldots \| \mathsf{ct}_{\mathsf{FHE}}^{(\alpha)}) \in \{0,1\}^\alpha$. Then, he generates

$$\mathsf{ct} := \left(\mathbf{s}^\top (\mathbf{A}_i + \mathsf{ct}_{\mathsf{FHE}}^{(i)} \cdot \mathbf{G}) + \mathbf{e}_i^\top\right)_{i \in [\alpha]},$$

where each $\mathbf{e}_i$ is sampled from a B-bounded error distribution. Note that Bob's encoding is independent of Alice's input length L.

Step 2: Simultaneous communication. Alice sends $\mathbf{A}_C$ to Bob, which is of size $m \cdot \mathsf{poly}(\lambda, d)$, and Bob sends ct to Alice, which is of size $\ell \cdot \mathsf{poly}(\lambda, d)$. Here, d denotes the circuit depth and is implicit in the LWE parameters.

Step 3: Local decoding. With the encodings from Step 1, Alice uses EvalCT to compute:

$$\begin{aligned}\mathbf{w}_C &:= \left(\mathbf{s}^\top (\mathbf{A}_{C_i} + C_i(\mathsf{ct}_{\mathsf{FHE}}) \cdot \mathbf{G}) + \tilde{\mathbf{e}}_i^\top\right)_{i \in [m]} \\ &= \left(\mathbf{s}^\top (\mathbf{A}_{C_i} + \mathsf{FHE.Enc}(\mathsf{sk}, f(X,y))[i] \cdot \mathbf{G}) + \tilde{\mathbf{e}}_i^\top\right)_{i \in [m]}.\end{aligned}$$

Then, because we set $\mathbf{s}[1] = 1$, the first coordinate of the i-th entry of $\mathbf{w}_C$ is:

$$(\mathbf{s}^\top \mathbf{A}_{C_i})[1] + C_i(y) + \tilde{\mathbf{e}}_i^\top[1] = (\mathbf{s}^\top \mathbf{A}_{C_i})[1] + \mathsf{FHE.Enc}(\mathsf{sk}, f(X,y))[i] + \tilde{\mathbf{e}}_i^\top[1]. \tag{1}$$

To see equality, it is helpful to note that the first column of the "gadget" matrix $\mathbf{G}$ (see Definition 12) is the vector $(1, 0, \ldots, 0)$. Moreover, by using a suitable FHE scheme, we can guarantee that FHE evaluation of a depth-d circuit can itself be evaluated by a bounded-depth circuit C.

We can view Equation (1) as a "noisy" share z_A of $\mathsf{FHE.Enc}(\mathsf{sk}, f(X,y))[i]$, since Bob can compute his corresponding "noisy" share as $z_B := (\mathbf{s}^\top \mathbf{A}_{C_i})[1]$ using his key $\mathbf{s}$ and the $\mathbf{A}_{C_i}$ he receives from Alice. With this, Alice and Bob now have "noisy shares" of the i-th bit of the ciphertext $\mathsf{FHE.Enc}(\mathsf{sk}, f(X,y))$, since

$$\begin{aligned}z_A - z_B &\approx \underbrace{(\mathbf{s}^\top \mathbf{A}_{C_i})[1] + \mathsf{FHE.Enc}(\mathsf{sk}, f(X,y))[i]}_{\text{Alice's share from Equation (1)}} - \underbrace{(\mathbf{s}^\top \mathbf{A}_{C_i})[1]}_{\text{Bob's share}} \\ &\approx \mathsf{FHE.Enc}(\mathsf{sk}, f(X,y))[i].\end{aligned}$$

Unfortunately, it is easy to observe that this does not get us any closer to obtaining an SMS scheme, since the two parties could just as easily have obtained a "trivial" secret sharing of $\mathsf{FHE.Enc}(\mathsf{sk}, f(X,y))[i]$ by having Alice compute the FHE evaluation directly (and setting z_A to the result) and Bob setting z_B to zero. Jumping ahead, we will resolve this by "non-interactively" decrypting the evaluated FHE ciphertext using extensions to the EvalPK and EvalCT algorithms. However, we first point out some additional problems we need to resolve with the above attempt:

(1) It does not result in the parties having shares of $f(X,y)$, rather they hold shares of the *encryption*, which is trivial to achieve using just FHE.
(2) The parties have *noisy* shares, which does not correspond to the additive-reconstruction we desire.
(4) It fails to provide privacy for Alice, since the algorithm EvalPK, as defined in [BGG+14], makes no guarantees about the privacy of the circuit C given $\mathbf{A}_C$ (recall that C contains Alice's input).
(3) While it does give input succinctness for Alice's input, it does not have output succinctness, since $|\mathbf{A}_C| = m \cdot \mathsf{poly}(\lambda, d)$ grows linearly with the size of the output m.

We now explain how we resolve these obstacles in our construction.

Obliviously decrypting the FHE evaluation. The first problem to address is that Alice and Bob obtain shares of the *encryption* of the result rather than a share of $f(X, y)$. Therefore, our first priority is letting Alice somehow "decrypt" the FHE ciphertext privately under Bob's secret key sk. Fortunately, this very problem was also faced in the predicate encryption scheme of Gorbunov et al. [GVW15]. In particular, they show that $(\mathsf{EvalPK}, \mathsf{EvalCT})$ can be "augmented" to compute circuits of the form $\mathsf{IP} \circ C$, where IP is the class of inner products. Moreover, while the input to C needs to be public (known to Alice), the inner product can be *private* (only known to Bob). Therefore, we can assume an FHE scheme with a near-linear decryption property and let Bob input the FHE secret decryption key sk as the private "inner product" input. With this modification, and by slightly abusing notation by using C_i to denote $\mathsf{IP} \circ C'_i$ (where C'_i is defined to output a vector in the ring $\mathbb{Z}_q$ corresponding to the FHE ciphertext encrypting the i-th bit of the circuit C), Alice and Bob obtain:

$$\underbrace{(\mathbf{s}^\top \mathbf{A}_{C_i})[1] + \overbrace{\langle \mathsf{FHE.Enc}(\mathsf{sk}, f_i(X,y)), \mathsf{sk} \rangle}^{(q/2)\cdot f_i(X,y)+\mathsf{noise}}}_{\text{Alice's share } z_A^{(i)}} - \underbrace{(\mathbf{s}^\top \mathbf{A}_{C_i})[1]}_{\text{Bob's share } z_B^{(i)}} \approx f_i(X,y),$$

where f_i computes the i-th bit of the function f. Note that this gives us a "noisy" secret sharing of the correct result! Indeed, the output of the circuit C consists of $m \cdot \beta$ ring elements, corresponding to the bit-wise encryptions of $f(X, y)$. Then, the inner product with the secret key $\mathsf{sk} \in \mathbb{Z}_q^\beta$ results in m noisy shares (over $\mathbb{Z}_q$) at the end, where the noise comes from the near-linear decryption.

Rounding of noisy shares. To convert these noisy shares to additive shares we can apply the "rounding lemma" [DHRW16, BKS19] to locally round-away the error:

Lemma 1 (Rounding of noisy secret shares [DHRW16, BKS19])**.** *Let (p, q) be two integers such that p divides q. Fix any $z \in \mathbb{Z}_q$ and let (z_0, z_1) be any two random elements of $\mathbb{Z}_q$ subject to $z_0 + z_1 = (q/p) \cdot z + e \bmod q$, where e is such that $q/(p \cdot |e|) \geq \lambda^{\omega(1)}$. Then, with probability at least $1 - (|e| + 1) \cdot p/q \geq 1 - \lambda^{-\omega(1)}$, it holds that $\lfloor z_0 \rceil_p + \lfloor z_1 \rceil_p = z \bmod p$, and the probability is over the random choice of $(z_0, z_1) \in \mathbb{Z}_q \times \mathbb{Z}_q$.*

Using the rounding lemma, and setting the LWE parameters of the FHE scheme to have a super-polynomial modulus-to-noise ratio, Alice and Bob locally derive shares $z_A^{(i)}$ and $z_B^{(i)}$, for all $i \in [m]$, such that $z_A^{(i)} - z_B^{(i)} = f_i(X, y)$, as desired. More concretely, we have:

$$\underbrace{\lfloor (\mathbf{s}^\top \mathbf{A}_{C_i})[1] + (q/2) \cdot f_i(X,y) + \mathsf{noise} \rceil_2}_{\text{Alice's share } z_A^{(i)}} - \underbrace{\lfloor (\mathbf{s}^\top \mathbf{A}_{C_i})[1] \rceil_2}_{\text{Bob's share } z_B^{(i)}} = f_i(X,y),$$

where noise corresponds to the sum of the FHE decryption and EvalCT errors.

Statistical hiding for Alice. The final problem we need to take care of is ensuring that $\mathbf{A}_C$ leaks no information about C (which contains Alice's input X). Fortunately, a related problem was faced by Quach et al. [QWW18] when constructing laconic function evaluation. They show an efficient transformation that can be applied to EvalPK (and EvalCT) such that $\mathbf{A}_C$ statistically hides C, which also hides Alice's input X in our construction.

Adding output succinctness. At this point we have a protocol that is input succinct: Bob's encoding is independent of f and only $\mathsf{poly}(\lambda, \ell)$ bits in size (which is independent of Alice's input size L). However, the encoding still grows with the output length $|f(X, y)|$, since $\mathbf{A}_C$ has to be at least as large as the output length m. To get output succinctness, we observe that we can cast Bob's computation as a vector oblivious linear evaluation (VOLE) [ADI+17]: Alice has input $\mathbf{A}_{C_i} \in \mathbb{Z}_q^{n \times k}$ and Bob has input $\mathbf{s} \in \mathbb{Z}_q^n$, and Bob needs to learn the linear evaluation $\mathbf{s}^\top \mathbf{A}_{C_i}$. It is therefore enough for Alice and Bob to compute *shares* of $\mathbf{s}^\top \mathbf{A}_{C_i}$ (in particular, Bob never needs to have $\mathbf{A}_{C_i}$ "in the clear"). To accomplish this, we "bootstrap" using using SMS for VOLE. That is, using an SMS scheme for a "batched" variant of VOLE, Alice encodes each matrix $\mathbf{A}_{C_i} \in \mathbb{Z}_q^{n \times k}$ (Alice has m such matrices) and Bob encodes his secret $\mathbf{s}$. Using the public encodings, the two parties then locally (without further communication) obtain additive shares of $\mathbf{s}^\top \mathbf{A}_{C_i}$ as output, for all $i \in [m]$. Moreover, using existing constructions of succinct, non-interactive VOLE [ARS24, BCM+24], which we show can be cast as an SMS scheme, the total communication of this protocol is $O(m^{2/3}) \cdot \mathsf{poly}(\lambda, d)$, giving us sublinearity in the output size.

2.2 Construction from $i\mathcal{O}$

We now overview our construction of SMS from $i\mathcal{O}$, where we realize SMS for polynomially-sized *batch* functions. In this setting, Alice has a vector of inputs $X = (x_1, \ldots, x_L)$ and Bob has an ℓ-bit input y. At a high level, our protocol is reminiscent of the *insecure* protocol sketched in Section 1, where Bob simply sends his input y to Alice. However, to provide privacy for Bob, we use $i\mathcal{O}$ to hide y inside an obfuscated program.

In more detail, the main idea is to have Bob send an obfuscated program for a universal circuit, with his short input y hardcoded in it. The program evaluates the function $f(x_i, y) \oplus R_i$, where f and x_i are given as input and R_i is a pseudorandom mask that depends on f and x_i. To allow Bob to compute R_i on his end, which becomes his share of $f(x_i, y)$, Alice commits to her inputs X in such a way that she can locally decommit to any input x_i later on. Then, R is computed as the output of a PRF on the commitment to the batch and the index i of the input x_i in the batch X. This allows Bob to locally derive his share of $f(x_i, y)$ without knowledge of Alice's inputs.

Concretely, we use SSB hashing [HW15], the standard tool to use in conjunction with $i\mathcal{O}$. The general template of $\mathsf{SSB} + i\mathcal{O}$ [HW15, OPWW15] is to hardcode the hash key hk and commitment c_X (informally, we will refer to the hash as a "commitment") into the obfuscated program. Then, when Alice runs the program, she can provide a local opening to x_i. Because the commitment is *statistically binding* at some index i, it becomes easy to prove security via a hybrid argument that switches out where the SSB hash is statistically vs. computationally binding. At a high level, this makes two adjacent hybrid programs functionally equivalent, which then enables invoking $i\mathcal{O}$ security to prove computational indistinguishability between the hybrids.

Unfortunately, we cannot directly apply this template to realize SMS. The problem is that Bob does not have Alice's commitment (hash) when he generates the obfuscated program at encoding time! At first, this problem appears insurmountable, because Alice (who gets the program that includes Bob's input y) can potentially extract y by running it with many different inputs $X' \neq X$ and learn information from the output (i.e., perform a resetting attack [HLP11]). Indeed, given that the program cannot check whether Alice correctly opens the x_i's relative to the hash she sent to Bob, it may appear that this approach is doomed to fail. This very problem underpins the impossibility result of Hubáček and Wichs [HW15].

We get around this, however, by leaning on the fact that the output to the parties are pseudorandom. That is, we only need to guarantee the output of Alice is a valid secret share of the output if she provides the correct commitment—even if Alice equivocates, she obtains a pseudorandom string that leaks no information on y.

We now turn to explaining our construction.

Setup. The setup consists of generating the public SSB hash key hk. For some specific instantiations of the construction, we can have hk be randomly distributed and hence only need a common random string setup (see Section 6).

Step 1: Generating the encodings. Alice computes a commitment to her large (batch) input X using an SSB hash function with the key hk. Alice's public encoding simply consists of the hash output that we denote as c_X. Bob, on the other hand, generates an obfuscation of a program P that has a PRF key K and his input y hardwired inside. This program takes as input *some* commitment c'_X (which may be different from c_X), a batch element x_i, an index i, an opening π_i (generated with respect to c'_X), and the description of a function f. The program first checks that $\mathsf{SSB.Verify}(\mathsf{hk}, c'_X, x_i, i, \pi_i) = 1$ (i.e., the opening is accepting) and then outputs $\mathcal{U}(f, x_i, y) \oplus R_i$, where $\mathcal{U}$ is the universal circuit and R_i is a pseudorandom mask computed as $F_K(c'_X \| f \| i)$. We emphasize that c'_X that is fed as input to the program need not match with c_X that Alice sent to Bob as her public encoding. Moreover, we stress that the PRF evaluation $F_K(c'_X \| f \| i)$ used to compute the output mask does not include the input x_i nor opening π_i, which are provided as input to the program. Jumping ahead, this will be necessary so as to let Bob recompute the mask R_i "on his side" at decoding time, without knowledge of (x_i, π_i).

Remark 1. We note that because the obfuscated program is generated for a universal circuit $\mathcal{U}$ that takes the function f as input, neither Alice nor Bob need to know the function f (except its size) when generating their respective encodings. This makes our $i\mathcal{O}$-based construction satisfy a stronger, function-adaptive variant of SMS, which we define formally in Section 4.

Step 2: Simultaneous communication. Alice sends c_X to Bob, which is of size $\mathsf{poly}(\lambda)$, and Bob sends the obfuscated program P to Alice, which is of size $\ell \cdot \mathsf{poly}(\lambda, \log L)$. In particular, the program P sketched above grows logarithmically with the batch size due to its dependence on the index i, for each input in the batch.

Step 3: Local decoding. For any function f (chosen adaptively at decoding time), Alice evaluates the obfuscated program on input (c'_X, x_i, i, π_i, f). Observe that if the verification passes, the program outputs $f(x_i, y) \oplus R_i$. Bob, on his end, computes $R_i := F_K(c_X \| f_i \| i)$, using his PRF key K. Observe that if $c_X = c'_X$, then Alice and Bob have shares of $f(x_i, y)$:

$$f(x_i, y) = \underbrace{(f(x_i, y) \oplus R_i)}_{\text{Alice's share}} \oplus \underbrace{R_i.}_{\text{Bob's share}}$$

They can repeat this process for all $i \in [L]$, and also any $f \in \mathcal{F}$, to obtain the shares of the function computed over all of Alice's L inputs $x_i \in X$.

Ideas behind the proof of security. As mentioned above, the commitment sent by Alice cannot be hardcoded into the program that is obfuscated by Bob. This makes the proof security more involved, since we need to consider all possible commitments that Alice can input into the program.

Let's start with Alice's security, which is the simpler case to analyze. Alice's message to Bob consists of an SSB hash of her private input. However, note that SSB hashing does *not* guarantee hiding of the input, making it possible to learn something about Alice's input message from the resulting hash. Our solution to this problem is to have Alice individually commit to each input (in her batch of inputs) using a regular, perfectly-binding and computationally-hiding commitment scheme, and then SSB hash the full set of commitments rather than her "raw" inputs. This still ensures a one-to-one mapping from inputs to the values hashed using the SSB hashing, and, in particular, preserves somewhere statistical binding. We modify the program sent by Bob to take a local opening to the SSB hash and also the opening to the underlying perfectly-binding commitment. With this modification, Alice's security reduces to the computational hiding of the commitment scheme.

Now we examine Bob's security. Bob's message to Alice consists of the obfuscated program P that has his private input y hardwired inside it. This program outputs $f(x_i, y) \oplus F_K(c'_X \| f \| i)$ if Alice is able to produce valid openings to x_i with respect to c'_X. If we assume that the obfuscation scheme satisfies VBB security [BGI+01], then the only thing that Alice can learn is this output. Because the output is masked with a PRF evaluation, this should intuitively provide no information about Bob's input.[10] However, as we detail below, care must be taken as the obfuscation only satisfies indistinguishability-based security.

To argue Bob's security, we change the obfuscated program P to a dummy program P^{sim} that just outputs $F_K(c'_X \| f \| i)$. Once we make this change, we can rely on the security of $i\mathcal{O}$ to remove the hardwired input y from the program. To switch the obfuscation from the real program to this dummy program, we rely on the punctured programming approach of Sahai and Waters [SW14]. Specifically, we consider a canonical ordering of the inputs $(c'_X \| f \| i)$ to the PRF and replace the obfuscated program to output a dummy PRF evaluation rather than the actual output, one input at a time, via a sequence of hybrids. But to make this change at a specific input $(c'_X \| f \| i)$, and switch from one hybrid to the next, we need to ensure that $F_K(c'_X \| f \| i)$ is used to mask only one output in the program P. This means that Alice should not be able to provide valid openings for two different x_i's with respect to the same hash c'_X. For this purpose, we rely on the somewhere binding property of the SSB hashing and the binding property of the underlying commitment scheme. Specifically, c'_X statistically determines Alice's input x_i if the SSB hashing is made to be binding at position i. Furthermore, the hiding property of SSB hashing allows us to switch the hashing key to be binding at location i without the adversary noticing this change. This allows us to make the security proof go through. Finally, because we have an exponential number of hybrids, we need to complexity leverage and rely on the sub-exponential security of $i\mathcal{O}$ and the PRF used to mask the outputs.

The full construction and proof are presented in Section 6.

[10] This requires a delicate argument as we must ensure that Alice can only provide valid openings to a single x_i for any i. This is argued using the somewhere binding property of the SSB hashing.

3 Preliminaries

3.1 Notation

In this section, we cover the notation that we will use throughout the paper.

Circuit and function classes. We define a class of circuits $\mathcal{C}$ to be a set of circuits, where each circuit $C \in \mathcal{C}$ has associated depth and size parameters. Unless otherwise stated, we will write $\mathcal{C} = \left\{C \colon \mathbb{Z}_q^\ell \to \mathbb{Z}_q^m\right\}$ to mean the set of all depth-d arithmetic (or Boolean in the case of $q = 2$) circuits of polynomial size that take ℓ inputs and produce m outputs in $\mathbb{Z}_q$. We will occasionally write C_i to indicate the circuit that computes the i-th output of a circuit C. Unless otherwise stated, we refer to a function family $\mathcal{F}$ as the set of functions represented by circuits in $\mathcal{C}$.

General notation. We let $\mathbb{N}$ denote the set of natural numbers. Unless otherwise stated, we will use $\mathsf{poly}(\cdot)$ to denote the set of all polynomials. We occasionally abuse notation and let poly denote a fixed polynomial.

Sampling and assignment. We let $x \leftarrow\!\!\$\, S$ denote a uniformly random sample drawn from S. We let $x \leftarrow \mathcal{A}$ denote assignment from a possibly randomized algorithm $\mathcal{A}$. We let $x := y$ denote initialization of x to the value of y.

Vectors and matrices. We denote a vector $\mathbf{v}$ using bold lowercase letters and a matrix $\mathbf{A}$ using bold uppercase letters. The i-th coordinate of a vector $\mathbf{v}$ is denoted by $\mathbf{v}[i]$. We will occasionally write $(v_i)_{i=1}^n$ to denote the vector $(v_1, \ldots, v_n)$. The i-th bit of a bit-string s is denoted by s_i.

Rounding. We let $\lfloor x \rceil$ denote the rounding of a real number x to the nearest integer. For integers $q \geq p \geq 2$, we define the modular rounding function $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lfloor v \rceil_p = \lfloor (p/q) \cdot v \rceil$.

Efficiency. By an *efficient* algorithm $\mathcal{A}$ we mean that $\mathcal{A}$ is modeled by a (possibly non-uniform) Turing Machine that runs in probabilistic polynomial time.

Indistinguishability. We write $D_0 \approx_c D_1$ to mean that two distributions D_0 and D_1 are *computationally* indistinguishable to all efficient distinguishers $\mathcal{D}$ and $D_0 \approx_s D_1$ to mean that D_0 and D_1 are *statistically* indistinguishable distributions.

3.2 The learning with errors assumption

Here, we recall the learning with errors (LWE) assumption [Reg05].

Definition 1 (Learning With Errors)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter and let χ_τ denote a discrete Gaussian distribution over $\mathbb{Z}_q$ with noise parameter τ. The learning with errors (LWE) assumption $\mathsf{LWE}_{n,k,q,\tau}$ holds for the parameters $n = n(\lambda)$, $k = k(\lambda)$, $q = q(\lambda)$, $\tau = \tau(\lambda)$ if*

$$\left(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top\right) \approx_c (\mathbf{A}, \mathbf{u}),$$

where $\mathbf{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{n \times k}$, $\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow\!\!\$\, \chi_\tau^k$, and $\mathbf{u} \leftarrow\!\!\$\, \mathbb{Z}_q^k$.

Fully homomorphic encryption. We recall here the definition of fully homomorphic encryption (FHE) [Gen09].

Definition 2 (Fully Homomorphic Encryption)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter and $\mathcal{M} = \mathcal{M}(\lambda)$ be a message space. A fully homomorphic encryption (FHE) scheme consists of four algorithms $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with the following syntax:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$. *The randomized key generation algorithm takes as input the security parameter λ. It outputs a public key pk and a secret key sk.*
- $\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$. *The randomized encryption algorithm takes as input the public key pk and a message $x \in \mathcal{M}$. It outputs a ciphertext ct.*
- $\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)) \to \mathsf{ct}'$. *The deterministic evaluation algorithm takes as input the public key pk, an ℓ-argument, m-output function f, and a tuple of ℓ ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ encrypting messages $(x_1, \ldots, x_\ell)$. It outputs a tuple of m evaluated ciphertexts $(\mathsf{ct}'_1, \ldots, \mathsf{ct}'_m)$.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to x$. *The deterministic decryption algorithm takes as input the secret key sk and a ciphertext ct. It outputs a message x.*

When Dec *takes as input a tuple of ciphertexts* $(\mathsf{ct}_1, \ldots, \mathsf{ct}_m)$ *it is understood to mean that* Dec *is applied individually to each ciphertext in the tuple.*

The above algorithms must satisfy the following properties:

Correctness. *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all sets of messages* $x_1, \ldots, x_\ell \in \mathcal{M}$ *and all* ℓ*-argument,* m*-output functions* f *that can be represented by a polynomial-size circuit, we have:*

$$\Pr\left[\begin{array}{c} \mathsf{Dec}(\mathsf{sk}, (\mathsf{ct}'_1, \ldots, \mathsf{ct}'_m)) \\ = f(x_1, \ldots, x_\ell) \end{array} \;:\; \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i), \forall i \in [\ell] \\ (\mathsf{ct}'_1, \ldots, \mathsf{ct}'_m) \leftarrow \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is over the randomness of KeyGen *and* Enc.

Compactness. *For all sets of messages* $x_1, \ldots, x_\ell \in \mathcal{M}$ *and all* ℓ*-argument,* m*-output functions* f*, it holds that:*

$$|\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)| = \mathsf{poly}(\lambda, m),$$

for some fixed polynomial $\mathsf{poly}(\cdot)$*. That is, the output length of* Eval *is independent of the input length* ℓ *and function description* f*.*

Security. *For all efficient adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l} b = b' \end{array} \;:\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pk}) \\ b \leftarrow\!\!\$\, \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b) \\ b' \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{ct}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Definition 3 (Secret-Key Fully Homomorphic Encryption)**.** *An FHE scheme* $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is a* secret-key *FHE scheme if it satisfies the syntax, correctness, and compactness properties of Definition 2 without the public key* pk*.*

4 Defining SMS Secure Computation

In this section, we first present the formal definition of SMS, some natural extensions of it, and discuss connections to other primitives.

Definition 4 (Simultaneous-Message and Succinct Secure Computation)**.** *Let* λ *be a security parameter,* $L = L(\lambda)$ *be the input length of Alice (who has the large input* X*),* $\ell = \ell(\lambda)$ *be the input length of Bob (who has the small input* y*),* $m = m(\lambda)$ *be the output length, where* L, ℓ, m *are all polynomial in* λ*, and let*

$$\mathcal{F} = \left\{ f \colon \{0,1\}^L \times \{0,1\}^\ell \to \{0,1\}^m \right\}$$

be a family of functions. A simultaneous-message and succinct (SMS) secure computation scheme for $\mathcal{F}$ *consists of a tuple of five efficient algorithms,*

$$\mathsf{SMS} = \big(\mathsf{Setup}, (\mathsf{Encode}_A, \mathsf{Decode}_A), (\mathsf{Encode}_B, \mathsf{Decode}_B)\big),$$

with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*. The randomized setup algorithm takes as input the security parameter and outputs a common reference string (CRS)* crs*.*
- $\mathsf{Encode}_\sigma(\mathsf{crs}, f, x) \to (\mathsf{pe}_\sigma, \mathsf{st}_\sigma)$*. The randomized encoding algorithm is parameterized by a party identifier* $\sigma \in \{A, B\}$*. It takes as input the CRS* crs*, a description of a function* $f \in \mathcal{F}$*, and an input* x*. It outputs a public encoding* pe_σ *and secret state* st_σ*.*
- $\mathsf{Decode}_\sigma(\mathsf{crs}, f, \mathsf{pe}_{\bar{\sigma}}, \mathsf{st}_\sigma) \to z_\sigma$*. The deterministic decoding algorithm is parameterized by a party identifier* $\sigma \in \{A, B\}$*. It takes as input the CRS* crs*, a description of a function* $f \in \mathcal{F}$*, the public encoding* $\mathsf{pe}_{\bar{\sigma}}$ *belonging to party* $\bar{\sigma} \neq \sigma$*, and secret state* st_σ *belonging to party* σ*. It outputs an* m*-bit string* $z_\sigma \in \{0,1\}^m$*.*

The above functionality must satisfy correctness, security, and succinctness:

Correctness. *For all security parameters $\lambda \in \mathbb{N}$, every pair of inputs $(X, y) \in \{0,1\}^L \times \{0,1\}^\ell$, and all functions $f \in \mathcal{F}$, an SMS scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that:*

$$\Pr\left[z_A \oplus z_B = f(X,y) \quad : \quad \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pe}_A, \mathsf{st}_A) \leftarrow \mathsf{Encode}_A(\mathsf{crs}, f, X) \\ (\mathsf{pe}_B, \mathsf{st}_B) \leftarrow \mathsf{Encode}_B(\mathsf{crs}, f, y) \\ z_A := \mathsf{Decode}_A(\mathsf{crs}, f, \mathsf{pe}_B, \mathsf{st}_A) \\ z_B := \mathsf{Decode}_B(\mathsf{crs}, f, \mathsf{pe}_A, \mathsf{st}_B) \end{array} \right] \geq 1 - \mathsf{negl}(\lambda).$$

Security. *For all efficient adversaries $\mathcal{A}$, for all $\sigma \in \{A, B\}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:*

$$\Pr\left[b = b' \quad : \quad \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{crs}) \\ b \leftarrow\$ \{0,1\} \\ (\mathsf{pe}_\sigma, \mathsf{st}_\sigma) \leftarrow \mathsf{Encode}_\sigma(\mathsf{crs}, x_b) \\ b' \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{pe}_\sigma) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

In words, the public encoding computationally hides the input of the party.

ϵ-Input Succinctness. *An SMS scheme is said to be* ϵ-input succinct*, for some $\epsilon \in [0,1)$, if for all security parameters $\lambda \in \mathbb{N}$, all $\sigma \in \{A, B\}$, every CRS crs, all inputs $X, y \in \{0,1\}^L \times \{0,1\}^\ell$, all output lengths m, and every pe_σ in the support of Encode_σ, it holds that*

$$|\mathsf{pe}_\sigma| \leq \mathsf{poly}(\lambda, \ell, m) \cdot L^\epsilon,$$

for some fixed polynomial poly*. In words, the public encoding generated by each party is sublinear in the size of the large input. If $\epsilon = 0$, then we say the SMS scheme is* fully input succinct*.*

We now define an additional (optional) property of output succinctness for SMS schemes. Similarly to the input succinctness property of Definition 4, output succinctness states that the encoding of each party must be sublinear in the function output length. In contrast to input succinctness—which is an integral property of Definition 4—the notion of SMS remains interesting even if the encodings are not succinct in the function output length. Indeed, output succinctness can, in some cases, be meaningless (e.g., when computing functions that output a single bit).

Definition 5 (ϵ-Output Succinctness)**.** *An SMS scheme is said to be* ϵ-output succinct*, for some $\epsilon \in [0,1)$, if for all security parameters $\lambda \in \mathbb{N}$, all $\sigma \in \{A, B\}$, every CRS crs, all inputs $X, y \in \{0,1\}^L \times \{0,1\}^\ell$, all output lengths $m \in \mathbb{N}$, and all pe_σ in the support of Encode_σ, it holds that*

$$|\mathsf{pe}_\sigma| \leq \mathsf{poly}(\lambda, L, \ell) \cdot m^\epsilon,$$

for some fixed polynomial $\mathsf{poly}(\cdot)$. In words, the public encoding generated by both parties is sublinear in the size of the output length. If $\epsilon = 0$, then we say the SMS scheme is fully output succinct*.*

Definition 6 (Succinctness)**.** *We say an SMS scheme is* ϵ-succinct *if it is both ϵ-input and ϵ-output succinct. If $\epsilon = 0$, we say it is* fully succinct*.*

Definition 7 (Function Adaptive)**.** *We say an SMS scheme is* function adaptive *if Encode_A and Encode_B take as input the function size $|f|$ in place of the function description f.*

Definition 8 (Additive Reconstruction)**.** *We say an SMS scheme has* additive *(resp.* subtractive*) reconstruction if the outputs of Decode_σ are defined over a finite Abelian group $\mathbb{G}$ (resp. over the integers) and the correctness property requires $z_A + z_B \in \mathbb{G}$ (resp. $z_A - z_B \in \mathbb{Z}$) equals $f(X, y)$.*

Definition 9 (Batch-Succinct SMS)**.** *Let $L, l, \ell, m \in \mathbb{N}$ be parameters of the SMS scheme. Let $\mathcal{F}$ be a family of batch functions, such that for each $f \in \mathcal{F}$, f takes a batch of inputs $X \in (\{0,1\}^l)^L$ and an input $y \in \{0,1\}^\ell$, and computes some function $g(X[i], y)$ with m-bit outputs, for each $i \in [L]$. An*

SMS scheme for the family $\mathcal{F}$ is said to be ϵ-batch-succinct *if for all security parameters $\lambda \in \mathbb{N}$, all $\sigma \in \{A, B\}$, every CRS* crs*, all input batches $X \in (\{0,1\}^l)^L$, and all inputs $y \in \{0,1\}^\ell$, it holds that*

$$|\mathsf{pe}_\sigma| \leq \mathsf{poly}(\lambda, \ell, l, m) \cdot L^\epsilon.$$

In words, the public encoding generated by each party grows sublinearly in the batch size.

Remarks on the definition of SMS. We provide some observations pertaining to Definition 4.

Remark 2 (Relation to a simulation-based definition). In Definition 4, we provide a game-based definition where the adversary must distinguish between encodings of two different adversarially-chosen messages. This definition is easier to work with and is conceptually simpler. In Appendix A, we prove that this game-based definition can be generically transformed into a simulation-based definition modeled by an ideal functionality.

Remark 3 (Common random string). We define the CRS as a common reference string for generality. In particular, some NIVOLE protocols (e.g., [ARS24, BCM+24]) satisfying Definition 4 have a structured common reference string. However, our constructions have a common *random* string.

Remark 4 (On input succinctness). We note that input succinctness for both parties *simultaneously* is information-theoretically impossible to achieve, as already observed by Abram et al. [ARS24] in the context of succinct homomorphic secret sharing. In particular, we cannot even have an *insecure* protocol satisfying input succinctness for both parties simultaneously.

Remark 5 (On output succinctness). Unlike input succinctness, *output succinctness* is only an interesting notion when satisfied for both parties. In particular, if only one of the public encodings is output-succinct, then the exchange of encodings will not necessarily be (i.e., if the information on the full output is present in one of the encodings).

Remark 6 (Post-composition with linear functions). Definition 4 can be used to compute functions of the form: $\mathbf{g} \otimes f(X, y)$, where the decoding algorithm additionally takes the linear transformation $\mathbf{g}$ as input. Note that such *post composition* by linear functions (e.g., see [BGI15]) is automatically implied by the additive reconstruction property of SMS.

4.1 Succinct, non-interactive VOLE as SMS

Here, we show how *succinct* non-interactive VOLE (NIVOLE) [ARS24, BCM+24] fits into our SMS definition. In a succinct NIVOLE scheme, Alice with a length-L input vector $\mathbf{a}$ and Bob with a scalar Δ (here, Bob's input length $\ell = 1$) compute additive shares of the vector $\Delta \cdot \mathbf{a}$, in a semi-honest, simultaneous-message protocol. Moreover, the succinctness property states that the communication of this single-round protocol is sublinear in L.

Definition 10 (Non-Interactive VOLE)**.** *We say that SMS instantiated with the function family $\mathcal{F} = \left\{ f_p \colon \mathbb{Z}_p^L \times \mathbb{Z}_p \to \mathbb{Z}_p^L \mid f_p \colon (\mathbf{a}, \Delta) \mapsto \Delta \cdot \mathbf{a} \right\}_{p \in \mathbb{N}}$, is an SMS scheme for non-interactive VOLE, denoted* NIVOLE*. We drop the subscript p from f_p when the ring $\mathbb{Z}_p$ is clear from context. We also omit the function f_p from Encode_σ and Decode_σ when the ring $\mathbb{Z}_p$ is fixed.*

Theorem 3 (Succinct NIVOLE from LWE [ARS24])**.** *For any integer p, assuming the hardness of LWE with a superpolynomial modulus-to-noise ratio, there exists an (2/3)-succinct scheme for NIVOLE instantiated over $\mathbb{Z}_p$.*

Batch non-interactive VOLE. We remark that we can view NIVOLE itself as a batch computation, since the same Δ is applied to all entries of Alice's large input $\mathbf{a}$. We define the following extension to NIVOLE which explicitly satisfies batch-SMS for NIVOLE.

Definition 11 (Batch NIVOLE)**.** *We define Batch NIVOLE for computing L matrix-vector product using the same vector. Concretely,* $\mathsf{BNIVOLE} = (\mathsf{Setup}, (\mathsf{Encode}_\sigma, \mathsf{Decode}_\sigma)_{\sigma \in \{A,B\}})$*, where* Encode_A *takes as input a list of matrices $(\mathbf{A}_i)_{i=1}^L$, where each $\mathbf{A}_i \in \mathbb{Z}_p^{\ell \times l}$, and* Encode_B *takes as input a vector $\mathbf{b} \in \mathbb{Z}_p^\ell$. Then, for all $\sigma \in \{A, B\}$,* Decode_σ *outputs an additive share of $(\mathbf{b}^\top \mathbf{A}_i)_{i=1}^L$.*

Lemma 2. *If there exists a succinct NIVOLE scheme with ϵ-succinctness, then there exists a batch NIVOLE scheme with ϵ-succinctness.*

Proof sketch. The construction is very simple: It suffices to run ℓ instances of NIVOLE in parallel, where Alice inputs the rows of the matrix $\mathbf{H} \in \mathbb{Z}_p^{(\ell \times l) \cdot L}$ consisting of matrices $(\mathbf{A}_i)_{i=1}^{L}$ concatenated together, and Bob inputs his vector $\mathbf{b}$. By the succinctness of NIVOLE, multiplying each entry of $\mathbf{b}$ by the corresponding row vector of $\mathbf{H}$ requires $(l \cdot L)^\epsilon$ communication (in particular, Alice's public encoding is sublinear in L). Then, by the post-composition with linear functions (cf. Remark 6), the columns of the resulting matrix can be summed together to obtain $\mathbf{b}^\top \mathbf{H}$. Moreover, this protocol satisfies Definition 9, since Alice's encoding is of size $\mathsf{poly}(\lambda, \ell) \cdot (l \cdot L)^\epsilon \leq \mathsf{poly}(\lambda, \ell, l) \cdot L^\epsilon$. ■

Remark 7. We note, in passing, that in the case of succinct NIVOLE, input and output succinctness are equivalent definitions, since the output length is identical to the input length of the party with the large input vector.

5 Construction from LWE

In this section, we present our LWE-based construction of SMS achieving both input and output succinctness.

5.1 Preliminaries

In this section, we present the necessary definitions and building blocks that we will use in our LWE-based construction of SMS.

Auxiliary functions. Here, we recall the algorithms EvalPK and EvalCT introduced in Boneh et al. [BGG+14] and the extensions of Gorbunov et al. [GVW15].

Definition 12 (Gadget Matrix [MP12])**.** *Let $q \geq 2$ be an integer. We call $\mathbf{g} := (1, 2, \ldots, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{\lceil \log q \rceil}$ the* gadget vector*. We call $\mathbf{G} := \mathbf{g} \otimes \mathbf{I}_n$ the* gadget matrix*.*

Definition 13 (Auxiliary Evaluation Algorithms [BGG+14,GVW15])**.** *Let α, β be integer parameters and let crs be a common random string (CRS). The auxiliary evaluation algorithms are two efficient and deterministic procedures $(\mathsf{EvalPK}, \mathsf{EvalCT})$ with the following syntax:*

- $\mathsf{EvalPK}(\mathsf{crs}, C) \to \mathbf{A}_{\mathsf{IP} \circ C}$*. Takes as input the CRS crs and a circuit $C\colon \{0,1\}^\alpha \to \mathbb{Z}_q^\beta$, and outputs a matrix $\mathbf{A}_{\mathsf{IP} \circ C} \in \mathbb{Z}_q^{n \times k}$.*
- $\mathsf{EvalCT}(\mathsf{crs}, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, C, x) \to \mathbf{w}_{\mathsf{IP} \circ C}$*. Takes as input the CRS crs, a list of vectors $(\mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta)$, the circuit C, and outputs a vector $\mathbf{w}_{\mathsf{IP} \circ C} \in \mathbb{Z}_q^k$.*

Lemma 3 (Adapted from [GVW15])**.** *Let $\lambda \in \mathbb{N}$ be a security parameter and $B = B(\lambda)$ be an integer bound. Under the $\mathsf{LWE}_{n,k,q,\tau}$ assumption with $k := n\lceil \log q \rceil$, there exist algorithms $(\mathsf{EvalPK}, \mathsf{EvalCT})$ satisfying Definition 13 for all integers $\alpha = \alpha(\lambda), \beta = \beta(\lambda)$ that are polynomial in the security parameter, such that for all common random strings of the form: $\mathsf{crs} := (\mathbf{A}_1, \ldots, \mathbf{A}_\alpha, \mathbf{B}_1, \ldots, \mathbf{B}_\beta) \in (\mathbb{Z}_q^{n \times k})^{\alpha + \beta}$, for all $\alpha + \beta$ vectors $\mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta \in \mathbb{Z}_q^k$, all $\mathbf{s} \in \mathbb{Z}_q^n$, all $(x, \mathbf{y}) \in \{0,1\}^\alpha \times \mathbb{Z}_q^\beta$, and all arithmetic circuits $C\colon \{0,1\}^\alpha \to \mathbb{Z}_q^\beta$ of depth d, if it holds that:*

$$\forall i \in [\alpha],\ \mathbf{u}_i = \mathbf{s}^\top (\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^\top \text{ and } ||\mathbf{e}_i||_\infty \leq B,$$
$$\forall i \in [\beta],\ \mathbf{v}_i = \mathbf{s}^\top (\mathbf{B}_i + \mathbf{y}[i] \cdot \mathbf{G}) + \mathbf{e}_i^\top \text{ and } ||\mathbf{e}_i||_\infty \leq B,$$

then it also holds that $\mathbf{w}_{\mathsf{IP} \circ C} := \mathsf{EvalCT}(\mathsf{crs}, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, C, x)$ is of the form:

$$\mathbf{w}_{\mathsf{IP} \circ C} = \mathbf{s}^\top (\mathbf{A}_{\mathsf{IP} \circ C} + \langle C(x), \mathbf{y} \rangle \cdot \mathbf{G}) + \mathbf{e}^\top \text{ with } ||\mathbf{e}||_\infty \leq (k+1)^d \cdot B,$$

where $\mathbf{A}_{\mathsf{IP} \circ C} := \mathsf{EvalPK}(\mathsf{crs}, C)$ and $\mathbf{G}$ is the gadget matrix from Definition 12.

Remark 8 (Public and private inputs). We stress that EvalCT does not take $\mathbf{y}$ as input. As such, we can view EvalCT as taking a "public" input x and "private" input $\mathbf{y}$ (encoded in the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\beta$), and using these to evaluate functions of the form $\langle C(x), \mathbf{y} \rangle$. Inspired by Gorbunov et al. [GVW15]'s approach for building predicate encryption, we will use this fact to let $\mathbf{y}$ be a secret decryption key that will allow Alice to obliviously decrypt an FHE ciphertext output by C.

Function-hiding. We now formalize the transformation used implicitly in the work of Quach et al. [QWW18] to make $\mathbf{A}_C$ (as output by EvalPK) statistically hiding.

Lemma 4 (Function-hiding Public Keys)**.** *Let γ be an integer. There exist efficient wrapper algorithms $\widetilde{\mathsf{EvalPK}}$ and $\widetilde{\mathsf{EvalCT}}$ defined as in Figure 1 such that:*

(1) $\widetilde{\mathsf{EvalPK}}$ and $\widetilde{\mathsf{EvalCT}}$ satisfy the properties defined in Lemma 3,

(2) if it holds that $\forall i \in [\gamma]$, $\mathbf{t}_i = \mathbf{s}^\top(\mathbf{C}_i + \mathbf{e}_i^\top)$, where $||\mathbf{e}_i||_\infty \leq B$, then it holds that the error magnitude in $\mathbf{w}_{\mathsf{IP}\circ C}$ (output by $\widetilde{\mathsf{EvalCT}}$) is at most an additive factor γB larger compared to the bound in Lemma 3, and

(3) if $\gamma \geq 2nk\lceil \log q \rceil$ and q is prime (see Lemma 3 for parameter details), then $\mathbf{A}_C$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{n\times k}$.

The transformation from Figure 1 is implicit in Appendix A of Quach et al. [QWW18]; we extract it here as a standalone wrapper for algorithms EvalPK and EvalCT.

QWW18 Function-Hiding Transformation

$\widetilde{\mathsf{EvalPK}}(\mathsf{crs}, C, \mathbf{C}_1, \ldots, \mathbf{C}_\gamma)$:

1 : $r \leftarrow\$ \{0,1\}^\gamma$
2 : $\mathbf{A}'_C \leftarrow \mathsf{EvalPK}(\mathsf{crs}, C)$
3 : $\mathbf{A}_C := \mathbf{A}'_C + \sum_{i=1}^{\gamma} r_i \mathbf{C}_i$
4 : **return** $(\mathbf{A}_C, r)$

$\widetilde{\mathsf{EvalCT}}(\mathsf{crs}, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, C, x, \mathbf{t}_1, \ldots, \mathbf{t}_\gamma, r)$:

1 : **parse** $r = r_1 \| \cdots \| r_\gamma$
2 : $\mathsf{ct}' \leftarrow \mathsf{EvalCT}(\mathsf{crs}, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, C, x)$
3 : $\mathbf{w}_{\mathsf{IP}\circ C} := \mathbf{w}'_{\mathsf{IP}\circ C} + \sum_{i=1}^{\gamma} r_i \mathbf{t}_i$
4 : **return** $\mathbf{w}_{\mathsf{IP}\circ C}$

Fig. 1. Function-hiding transformation of Quach et al. [QWW18].

Theorem 4 (FHE with Near-linear Decryption [BV11, BGV12, GSW13])**.** *Under the $\mathsf{LWE}_{n,k,q,\tau}$ assumption (cf. Definition 1), there exists a fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ for computing depth-d circuits, where the secret keys are vectors in $\mathbb{Z}_q^\beta$ and where the evaluation algorithm $\mathsf{FHE.Eval}$ outputs a vector $\mathsf{ct} \in \mathbb{Z}_q^\beta$ such that, for the corresponding secret key $\mathbf{s}$, it holds that (with probability 1):*

$$\langle \mathsf{ct}, \mathbf{s} \rangle = \lfloor q/p \rceil \cdot \mathsf{FHE.Dec}(\mathbf{s}, \mathsf{ct}) + e \bmod q$$

for some error $e \in \mathbb{Z}_q$ and where $|e| < (k+1)^d \cdot \mathsf{poly}(\lambda)$. Under the circular-security of the $\mathsf{LWE}_{n,k,q,\tau}$ assumption, FHE can be used to compute all circuits in P/poly such that $|e| < (k+1) \cdot \mathsf{poly}(\lambda)$. Moreover, for all circuits C of depth d, $\mathsf{FHE.Eval}(C, \cdot)$ can itself be computed by a circuit of depth $d \cdot \mathsf{polylog}(\lambda)$ and size $|C| \cdot \mathsf{poly}(\lambda, d)$.

5.2 Construction

Our construction is presented in Figures 2 to 4 and closely follows the technical overview.

5.3 Setting the parameters

For the LWE-based construction, we make use of Lemma 3 to evaluate an FHE evaluation on a ciphertext. In the construction, the circuit C computes $\mathsf{FHE.Eval}(f, \cdot)$ on some input ciphertext ct, where f is a function that is represented by a depth-d circuit. Therefore, using Theorem 4, the circuit C must have depth $d' \geq d \cdot \mathsf{polylog}(\lambda) \in \mathsf{poly}(\lambda, d)$.

We can set the LWE parameters n, k, q, τ, required for Lemma 3 and Theorem 5, as follows. Let $n = \mathsf{poly}(\lambda)$ and let B be an integer bound on the noise distribution determined by the parameter τ. We let $k = n\lceil \log q \rceil$. Then, for correctness, we need $q > 2^{\mathsf{poly}(\lambda, d')}$, for some polynomial $\mathsf{poly}(\cdot)$, subject to $q > 4 \cdot (k+1)^{d'} \cdot B \cdot \lambda^{\omega(1)}$.

SMS from LWE

Public Parameters. Let m be the function output length, n, k, q, χ_τ be LWE parameters (cf. Definition 1) as required by Lemma 3 and Theorem 4, $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a secret-key FHE scheme satisfying Definition 3, $F\colon \{0,1\}^\lambda \times [m] \to \mathbb{Z}_q$ be a PRF, α be the length (in bits) of an FHE ciphertext encrypting ℓ bits, β be the length (in elements of $\mathbb{Z}_q$) of the FHE secret key from Theorem 4, and γ be as required in Lemma 4.

$\mathsf{SMS.Setup}(1^\lambda)$:

1 : $\mathsf{crs}_{\mathsf{aux}} := (\mathbf{A}_1, \ldots, \mathbf{A}_\alpha, \mathbf{B}_1, \ldots, \mathbf{B}_\beta) \leftarrow\$ (\mathbb{Z}_q^{n\times k})^{\alpha+\beta}$

2 : $\mathsf{crs}_{\mathsf{rnd}} := (\mathbf{C}_1, \ldots, \mathbf{C}_\gamma) \leftarrow\$ (\mathbb{Z}_q^{n\times k})^{\gamma}$

3 : $\mathsf{crs}_{\mathsf{bvole}} \leftarrow \mathsf{BNIVOLE.Setup}(1^\lambda)$ // See Theorem 3.

4 : $K \leftarrow\$ \{0,1\}^\lambda$ // PRF key for randomizing output shares.

5 : **return** $\mathsf{crs} := (\mathsf{crs}_{\mathsf{aux}}, \mathsf{crs}_{\mathsf{rnd}}, \mathsf{crs}_{\mathsf{bvole}}, K)$

– See Figure 3 for the description of the encoding algorithms $\mathsf{SMS.Encode}_\sigma$.
– See Figure 4 for the description of the decoding algorithms $\mathsf{SMS.Decode}_\sigma$.

Fig. 2. Simultaneous-Message Succinct Secure Computation from LWE.

5.4 Security analysis

Here, we analyze the correctness and security of the SMS construction from Figure 2. We prove the following theorem.

Theorem 5. *Let λ be a security parameter and $d = d(\lambda) \in \mathsf{poly}(\lambda)$ be a circuit depth. Assume that the LWE assumption holds with a superpolynomial modulus-to-noise ratio. Then, Figure 2 is an SMS scheme satisfying Definition 4 for all functions that can be represented by polynomial-size, depth-d circuits. The scheme achieves full input succinctness and $(2/3)$-output succinctness (cf. Definition 5).*

Proof. We prove each required property in turn.

Correctness. We argue correctness for the i-th output bit, for any $i \in [m]$. By construction, we have that $\mathbf{z}_A^{(i)}$, as computed by Alice, is:

$$
\begin{aligned}
\mathbf{w}_{\mathsf{IP}\circ C_i} &:= \widetilde{\mathsf{EvalCT}}(\mathsf{crs}, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, C_i, \mathsf{ct}_y, \mathbf{t}_1, \ldots, \mathbf{t}_\gamma, r_i) \\
&= \mathbf{s}^\top(\mathbf{A}_C + \langle C_i(\mathsf{ct}_y), \mathsf{sk}\rangle \cdot \mathbf{G}) + \mathbf{e}^\top \qquad \text{// Follows from Lemmas 3 and 4.} \\
&= \mathbf{s}^\top (\mathbf{A}_C + \langle \mathsf{FHE.Eval}(f_i, (X, \mathsf{ct}_y)), \mathsf{sk}\rangle \cdot \mathbf{G}) + \mathbf{e}^\top \qquad \text{// Definition of } C_i. \\
&= \mathbf{s}^\top (\mathbf{A}_C + \langle \mathsf{FHE.Enc}(\mathsf{sk}, f_i(X, y)), \mathsf{sk}\rangle \cdot \mathbf{G}) + \mathbf{e}^\top \qquad \text{// Correctness of FHE.} \\
&= \mathbf{s}^\top \left(\mathbf{A}_C + \left(f_i(X, y)\frac{q}{2} + \mathbf{e}'^\top\right) \cdot \mathbf{G}\right) + \mathbf{e}^\top \qquad \text{// Near-linear decryption (cf. Theorem 4).} \\
&= \mathbf{s}^\top \mathbf{A}_C + \mathbf{s}^\top \left(\left(f_i(X, y)\frac{q}{2} + \mathbf{e}'^\top\right) \cdot \mathbf{G}\right) + \mathbf{e}^\top,
\end{aligned}
$$

where f_i is the function that computes the i-th bit of f.

The first equality follows directly from the correctness of $\widetilde{\mathsf{EvalCT}}$, as defined in Lemmas 3 and 4. The second equality follows from the definition of the circuit C_i, the properties of $\mathsf{FHE.Eval}$ from Theorem 4, and choice of parameters in Section 5.3. The third equality follows from the correctness of the FHE scheme and the fact that X can, without loss of generality, be converted to a ciphertext by having ct_y contain auxiliary encryptions of 0 and 1 (allowing any evaluator to convert their plaintext input into ciphertexts encrypted under the secret key). The fourth equality follows from the near-linear decryption property of the FHE scheme.

Then, because we set $\mathbf{s}[1] = 1$, we have that $\mathbf{w}_{\mathsf{IP}\circ C_i}[1] = (\mathbf{s}^\top \mathbf{A}_C)[1] + f_i(X, y)\frac{q}{2} + e' + e$, where e' and e are the first entries of $\mathbf{e}'$ and $\mathbf{e}$, respectively. To see this, it is helpful to note that the first column of $\mathbf{G}$ is of the form $(1, 0, \ldots, 0)^\top$; see Definition 12.

SMS from LWE: Encoding Algorithms

$\mathsf{SMS.Encode}_A(\mathsf{crs}, f, X)$:

parse $\mathsf{crs} = (\mathsf{crs}_{\mathsf{aux}}, \mathsf{crs}_{\mathsf{rnd}}, \mathsf{crs}_{\mathsf{bvole}}, K)$
Define an arithmetic circuit $C(\cdot)$ computing $\mathsf{FHE.Eval}(f, (X, \mathsf{ct}))$ on input $\mathsf{ct} \in \{0,1\}^\alpha$, where X is hardcoded as an input in C.
Let C_i be the circuit that computes the i-th output bit of C.
foreach $i \in [m]$:
  $(\mathbf{A}_{C_i}, r_i) \leftarrow \widetilde{\mathsf{EvalPK}}(\mathsf{crs}_{\mathsf{aux}}, C_i, \mathsf{crs}_{\mathsf{rnd}})$ // See Lemma 4.
$(\mathsf{pe}_A^{\mathsf{bvole}}, \mathsf{st}_A^{\mathsf{bvole}}) \leftarrow \mathsf{BNIVOLE.Encode}_A(\mathsf{crs}_{\mathsf{bvole}}, (\mathbf{A}_{C_i})_{i=1}^m)$ // See Definition 11.
$\mathsf{pe}_A := \mathsf{pe}_A^{\mathsf{bvole}}, \quad \mathsf{st}_A := (\mathsf{st}_A^{\mathsf{bvole}}, r_1, \ldots, r_m)$
return $(\mathsf{pe}_A, \mathsf{st}_A)$

$\mathsf{SMS.Encode}_B(\mathsf{crs}, f, y)$:

parse $\mathsf{crs} = (\mathsf{crs}_{\mathsf{aux}}, \mathsf{crs}_{\mathsf{rnd}}, \mathsf{crs}_{\mathsf{bvole}}, K)$
$\mathbf{s} \leftarrow\$ \mathbb{Z}_q^n$ subject to $\mathbf{s}[1] = 1$
$\mathsf{sk} \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$
$\mathsf{ct}_y \leftarrow \mathsf{FHE.Enc}(\mathsf{sk}, y)$
parse $\mathsf{ct}_y = c_1 \| c_2 \| \cdots \| c_\alpha \in \{0,1\}^\alpha$
foreach $i \in [\alpha]$:
  $\mathbf{e}_i \leftarrow\$ \chi_\tau^k, \quad \mathbf{u}_i := \mathbf{s}^\top (\mathbf{A}_i + c_i \cdot \mathbf{G}) + \mathbf{e}_i^\top$ // Bit-wise encryptions of ct_y.
parse $\mathsf{sk} = \mathsf{sk}_1 \| \mathsf{sk}_2 \| \cdots \| \mathsf{sk}_\beta \in \mathbb{Z}_q^\beta$ // See Theorem 4.
foreach $i \in [\beta]$:
  $\mathbf{e}'_i \leftarrow\$ \chi_\tau^k, \quad \mathbf{v}_i := \mathbf{s}^\top (\mathbf{B}_i + \mathsf{sk}_i \cdot \mathbf{G}) + \mathbf{e}_i'^\top$
foreach $i \in [\gamma]$:
  $\mathbf{e}''_i \leftarrow\$ \chi_\tau^k, \quad \mathbf{t}_i := \mathbf{s}^\top \mathbf{C}_i + \mathbf{e}_i''^\top$
$(\mathsf{pe}_B^{\mathsf{bvole}}, \mathsf{st}_B^{\mathsf{bvole}}) \leftarrow \mathsf{BNIVOLE.Encode}_B(\mathsf{crs}_{\mathsf{bvole}}, \mathbf{s})$
$\mathsf{pe}_B := (\mathsf{ct}_y, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, \mathbf{t}_1, \ldots, \mathbf{t}_\gamma, \mathsf{pe}_B^{\mathsf{bvole}}), \quad \mathsf{st}_B := \mathsf{st}_B^{\mathsf{bvole}}$
return $(\mathsf{pe}_B, \mathsf{st}_B)$

Fig. 3. Encoding algorithms for SMS from LWE (Figure 2).

Then, by correctness of BNIVOLE, we have that:

$$\begin{aligned}&\mathsf{BNIVOLE.Decode}_A(\mathsf{crs}_{\mathsf{bvole}}, \mathsf{pe}_B^{\mathsf{bvole}}, \mathsf{st}_A^{\mathsf{bvole}})[i] \\ &\quad - \mathsf{BNIVOLE.Decode}_B(\mathsf{crs}_{\mathsf{bvole}}, \mathsf{pe}_A^{\mathsf{bvole}}, \mathsf{st}_B^{\mathsf{bvole}})[i] = \mathbf{d}_A^{(i)} - \mathbf{d}_B^{(i)} = \mathbf{s}^\top \mathbf{A}_{C_i},\end{aligned}$$

with all but negligible probability.

Therefore, we have that with all but negligible probability, $\mathbf{z}_A^{(i)}$ and $\mathbf{z}_B^{(i)}$, as computed in $\mathsf{SMS.Decode}_A$ and $\mathsf{SMS.Decode}_B$, respectively, satisfy:

$$\begin{aligned}\mathbf{z}_A^{(i)} - \mathbf{z}_B^{(i)} &= \mathbf{d}_A^{(i)} + \mathbf{w}_{\mathsf{IP} \circ C_i} - \mathbf{d}_B^{(i)} \\ &= \mathbf{w}_{\mathsf{IP} \circ C_i} - \mathbf{s}^\top \mathbf{A}_{C_i} \\ &= \left(\mathbf{s}^\top \mathbf{A}_{C_i} + \mathbf{s}^\top \left(\left(f_i(X,y)\frac{q}{2} + \mathbf{e}'^\top\right) \cdot \mathbf{G}\right) + \mathbf{e}^\top\right) - \mathbf{s}^\top \mathbf{A}_{C_i}.\end{aligned}$$

And so it holds that, with all but negligible probability,

$$\mathbf{z}_A^{(i)}[1] - \mathbf{z}_B^{(i)}[1] = \mathbf{w}_{\mathsf{IP} \circ C_i}[1] - \mathbf{s}^\top \mathbf{A}_{C_i}[1] = f_i(X,y)\frac{q}{2} + e' + e.$$

SMS from LWE: Decoding Algorithms

$\mathsf{SMS.Decode}_A(\mathsf{crs}, f, \mathsf{pe}_B, \mathsf{st}_A)$:

1 : **parse** $\mathsf{crs} = (\mathsf{crs}_{\mathsf{aux}}, \mathsf{crs}_{\mathsf{rnd}}, \mathsf{crs}_{\mathsf{bvole}}, K)$

2 : **parse** $\mathsf{pe}_B = (\mathsf{ct}_y, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, \mathbf{t}_1, \ldots, \mathbf{t}_\gamma, \mathsf{pe}_B^{\mathsf{bvole}})$

3 : **parse** $\mathsf{st}_A = (\mathsf{st}_A^{\mathsf{bvole}}, r_1, \ldots, r_m)$

4 : **foreach** $i \in [m]$:

5 : $\quad \mathbf{w}_{\mathsf{IP} \circ C_i} \leftarrow \widetilde{\mathsf{EvalCT}}(\mathsf{crs}, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, C_i, \mathsf{ct}_y, \mathbf{t}_1, \ldots, \mathbf{t}_\gamma, r_i)$

6 : $(\mathbf{d}_A^{(i)})_{i=1}^m \leftarrow \mathsf{BNIVOLE.Decode}_A(\mathsf{crs}_{\mathsf{bvole}}, \mathsf{pe}_B^{\mathsf{bvole}}, \mathsf{st}_A^{\mathsf{bvole}})$

7 : **foreach** $i \in [m]$:

8 : $\quad z_A^{(i)} := \left\lfloor (\mathbf{d}_A^{(i)} + \mathbf{w}_{\mathsf{IP} \circ C_i})[1] + F_K(i) \right\rceil_2$

9 : **return** $(z_A^{(1)}, \ldots, z_A^{(m)})$

$\mathsf{SMS.Decode}_B(\mathsf{crs}, f, \mathsf{pe}_A, \mathsf{st}_B)$:

1 : **parse** $\mathsf{crs} = (\mathsf{crs}_{\mathsf{aux}}, \mathsf{crs}_{\mathsf{rnd}}, \mathsf{crs}_{\mathsf{bvole}}, K)$

2 : **parse** $\mathsf{pe}_A = \mathsf{pe}_A^{\mathsf{bvole}}$ **and** $\mathsf{st}_B = \mathsf{st}_B^{\mathsf{bvole}}$

3 : $(\mathbf{d}_B^{(i)})_{i=1}^m \leftarrow \mathsf{BNIVOLE.Decode}_B(\mathsf{crs}_{\mathsf{bvole}}, \mathsf{pe}_A^{\mathsf{bvole}}, \mathsf{st}_B^{\mathsf{bvole}})$

4 : **foreach** $i \in [m]$:

5 : $\quad z_B^{(i)} := \left\lfloor \mathbf{d}_B^{(i)}[1] + F_K(i) \right\rceil_2$

6 : **return** $(z_B^{(1)}, \ldots, z_B^{(m)})$

Fig. 4. Decoding algorithms for SMS from LWE (Figure 2).

Importantly, we have that e' (the FHE decryption error) and e (the EvalCT evaluation error) are bounded in magnitude by Lemma 3 and Theorem 4. Specifically, we have that $\max(|e|, |e'|) \leq (k+1)^{d'} \cdot \mathsf{poly}(\lambda)$, and so it holds that $|e + e'| \leq 2(k+1)^{d'} \cdot \mathsf{poly}(\lambda)$, where $d' > d \cdot \mathsf{polylog}(\lambda)$. Therefore, if we have that $q > 4(k+1)^{d'} \cdot \lambda^{\omega(1)}$ (which necessitates assuming LWE security holds with a superpolynomial modulus-to-noise ratio), then by Lemma 1 and the above analysis, we get that:

$$\Pr\Big[\left\lfloor \mathbf{z}_A^{(i)}[1] + F_K(i) \right\rceil_2 - \left\lfloor \mathbf{z}_B^{(i)}[1] + F_K(i) \right\rceil_2 = \left\lfloor \mathbf{z}_A^{(i)}[1] - \mathbf{z}_B^{(i)}[1] \right\rceil_2 = f_i(X, y) \Big] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is over randomness of $\mathbf{s}$ and the PRF key K. In particular, the PRF ensures a pseudorandom distribution over $\mathbb{Z}_q$ (and is equivalent to randomizing the subtractive shares), which then allows us to apply Lemma 1. It follows that the outputs of $\mathsf{SMS.Encode}_A$ and $\mathsf{SMS.Encode}_B$ form subtractive shares of all m output bits of $f(X, y)$, with all but negligible probability in λ.

This concludes the proof of correctness.

Succinctness. We now briefly analyze the input and output succinctness. For input succinctness, observe that Alice's encoding pe_A consists only of the $\mathsf{BNIVOLE}$ public encoding of the batch NIVOLE SMS scheme involving m matrices, where each matrix is of size $\mathsf{poly}(\lambda, d)$ by Definition 13. In particular, we get full *input* succinctness, since the size of pe_A is independent of Alice's input length $|X|$ and only depends on the output length m.

Then, for output succinctness, by Lemma 2 and Theorem 3 we have that $|\mathsf{st}_A^{\mathsf{bvole}}| \leq \mathsf{poly}(\lambda, d) \cdot m^\epsilon$ with $\epsilon = (2/3)$, where d is the circuit depth and is implicit in the choice of modulus q. Moreover, Bob's encoding pe_B consists of the batch NIVOLE public encoding, which has size $\mathsf{poly}(\lambda, d) \cdot m^\epsilon$ with $\epsilon = (2/3)$ along with (1) an FHE ciphertext encrypting the input y under the secret key sk and (2) $(\alpha + \beta + \gamma)$ ciphertexts encrypted under $\mathbf{s}$. These ciphertexts all have, at most, a linear dependence on $|y|$, and thus have a combined length of $|y| \cdot \mathsf{poly}(\lambda, d)$. Thus, Alice's and Bob's encoding are both sublinear in the output length m, which proves the ϵ output succinctness property.

Security for Alice. We show that Alice's encoding reveals no information on her private input X. Note that Alice's public encoding pe_A consists of $\mathbf{A}_C$ computed according to $\widetilde{\mathsf{EvalPK}}(\mathsf{crs}, C, \mathbf{C}_1, \ldots, \mathbf{C}_\gamma)$, where C has X hardcoded inside it. By Lemma 4, we have that $\mathbf{A}_C$ is statistically close to uniform, and so indistinguishability holds trivially.

Security for Bob. We prove that the real view of the adversary is computationally indistinguishable to a simulated view. First, we construct the following efficient simulator $\mathcal{S}$ for pe_B (Bob's public encoding from Figure 2).

$\mathcal{S}$: On input crs,
- Parse $\mathsf{crs} = (\_, \_, \mathsf{crs}_{\mathsf{bvole}})$.
- Sample $\mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta$, and $\mathbf{t}_1, \ldots, \mathbf{t}_\gamma$ uniformly at random.
- Sample $\mathsf{sk} \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$.
- Compute $\mathsf{ct}_y \leftarrow \mathsf{FHE.Enc}(\mathsf{sk}, 0)$.
- Compute $(\mathsf{pe}_B^{\mathsf{bvole}}, \_) \leftarrow \mathsf{BNIVOLE.Encode}_B(\mathsf{crs}_{\mathsf{bvole}}, 0)$.
- Output $\mathsf{pe}_B := (\mathsf{ct}_y, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, \mathbf{t}_1, \ldots, \mathbf{t}_\gamma, \mathsf{pe}_B^{\mathsf{bvole}})$.

We prove that the output of $\mathcal{S}$ is computationally indistinguishable from the real view under the LWE assumption using a hybrid argument.

- *Hybrid* $\mathcal{H}_0$. This hybrid consists of pe_B computed exactly according to Encode_B in Figure 2.
- *Hybrid* $\mathcal{H}_1$. In this hybrid, we replace $\mathsf{pe}_B^{\mathsf{bvole}}$ in pe_B with the encoding of zero. That is, $\mathsf{pe}_B^{\mathsf{bvole}}$ is generated according to $\mathsf{BNIVOLE.Encode}_B(\mathsf{crs}_{\mathsf{bvole}}, 0)$.

  Claim. $\mathcal{H}_0 \approx_c \mathcal{H}_1$ under the LWE assumption.

  Proof. Computational indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$ follows immediately from the security of $\mathsf{BNIVOLE}$, which is realized under LWE with a superpolynomial modulus-to-noise ratio (cf. Theorem 3). □

- *Hybrid* $\mathcal{H}_2$. In this hybrid, we make $\mathbf{s}$ uniformly random in $\mathbb{Z}_q^n$ and no longer set $\mathbf{s}[1] = 1$.

  Claim. $\mathcal{H}_1 \approx_c \mathcal{H}_2$ under the LWE assumption.

  Proof. Computational indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ follows immediately from the leakage-resilience of the LWE assumption [GKPV10]; in particular, the LWE assumption is tolerant to any constant number of coordinates of the secret being set to a fixed public value. □

- *Hybrid* $\mathcal{H}_3$. In this hybrid, we replace $\mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, \mathbf{t}_1, \ldots, \mathbf{t}_\beta$ in pe_B (which all depend on the secret $\mathbf{s}$) with uniformly random values sampled independently from $\mathbb{Z}_q^k$.

  Claim. $\mathcal{H}_2 \approx_c \mathcal{H}_3$ under the LWE assumption.

  Proof. Suppose, towards contradiction, that $\mathcal{H}_2 \not\approx_c \mathcal{H}_3$, then there exists an efficient distinguisher $\mathcal{A}$ distinguishing between $\mathcal{H}_2$ and $\mathcal{H}_3$ with non-negligible advantage.
  Let $k' := \alpha + \beta + \gamma$. We construct an efficient distinguisher $\mathcal{B}$ for the LWE problem that receives as input $k' = k'(\lambda) \in \mathsf{poly}(\lambda)$ LWE challenge samples $(\mathbf{R}_i, \mathbf{r}_i)_{i=1}^{k'}$, where $(\mathbf{r}_1, \ldots, \mathbf{r}_{k'})$ are either all uniformly random and independent or distributed as $\mathbf{s}^\top \mathbf{R}_i + \mathbf{e}_i$, for all $i \in [k']$.
  $\mathcal{B}$ proceeds as follows:
  1. Sample $\mathsf{sk} \leftarrow\!\!\$\, \mathsf{FHE.KeyGen}(1^\lambda)$ and computes ct_y exactly as in Figure 2.
     Let $\mathsf{sk} := (\mathsf{sk}_1, \ldots, \mathsf{sk}_\beta)$ and $\mathsf{ct}_y = c_1 \| c_2 \| \cdots \| c_\alpha \in \{0,1\}^\alpha$.
  2. For each $i \in [\alpha]$, set $\mathbf{u}_i := \mathbf{r}_i$ and set $\mathbf{A}_i := \mathbf{R}_i - c_i \mathbf{G}$.
  3. For each $i \in [\beta]$, set $\mathbf{v}_i := \mathbf{r}_{\alpha+i}$ and set $\mathbf{B}_i := \mathbf{R}_{\alpha+i} - \mathsf{sk}_i \mathbf{G}$.
  4. For each $i \in [\gamma]$, set $\mathbf{t}_i := \mathbf{r}_{\alpha+\beta+i}$ and $\mathbf{C}_i := \mathbf{R}_{\alpha+\beta+i}$.

5. Set $\mathsf{crs}_{\mathsf{aux}} := (\mathbf{A}_1, \ldots, \mathbf{A}_\alpha, \mathbf{B}_1, \ldots, \mathbf{B}_\beta)$, $\mathsf{crs}_{\mathsf{rnd}} := (\mathbf{C}_1, \ldots, \mathbf{C}_\gamma)$.
6. Set $\mathsf{crs} := (\mathsf{crs}_{\mathsf{aux}}, \mathsf{crs}_{\mathsf{rnd}}, \mathsf{crs}_{\mathsf{bvole}}, K)$, where $\mathsf{crs}_{\mathsf{rnd}}$, $\mathsf{crs}_{\mathsf{bvole}}$ and K are sampled as in Figure 2.
7. Set $\mathsf{pe}_B := (\mathsf{ct}_y, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, \mathbf{t}_1, \ldots, \mathbf{t}_\gamma, \mathsf{pe}_B^{\mathsf{bvole}})$, where $\mathsf{pe}_B^{\mathsf{bvole}}$ is distributed exactly as in $\mathcal{H}_2$.
8. Output as $\mathcal{A}(\mathsf{crs}, \mathsf{pe}_B)$ does.

We now argue that $\mathcal{B}$ wins with the same advantage as $\mathcal{A}$. Observe that when $\mathcal{B}$ receives LWE samples $(\mathbf{r}_1, \ldots, \mathbf{r}_{k'})$, then we have that:

- $\mathbf{u}_i = \mathbf{r}_i = \mathbf{s}^\top \mathbf{R}_i + \mathbf{e}_i = \mathbf{s}^\top (\mathbf{A}_i + c_i \mathbf{G}) + \mathbf{e}_i$, for all $i \in [\alpha]$.
- $\mathbf{v}_i = \mathbf{r}_{\alpha+i} = \mathbf{s}^\top \mathbf{R}_{\alpha+i} + \mathbf{e}_{\alpha+i} = \mathbf{s}^\top (\mathbf{B}_i + \mathsf{sk}_i \mathbf{G}) + \mathbf{e}_{\alpha+i}$, for all $i \in [\beta]$.
- $\mathbf{t}_i = \mathbf{r}_{\alpha+\beta+i} = \mathbf{s}^\top \mathbf{R}_{\alpha+\beta+i} + \mathbf{e}_{\alpha+\beta+i} = \mathbf{s}^\top \mathbf{C}_i + \mathbf{e}_{\alpha+\beta+i}$, for all $i \in [\gamma]$.

and so $\mathcal{A}$ receives pe_B distributed identically to hybrid $\mathcal{H}_2$.

In contrast, if $\mathcal{B}$ receives uniformly random samples $(\mathbf{r}_1, \ldots, \mathbf{r}_{k'})$, then all $\mathbf{u}_i$, $\mathbf{v}_i$, $\mathbf{t}_i$ are uniformly random, which is distributed identically to hybrid $\mathcal{H}_3$. Therefore, $\mathcal{B}$ succeeds with the same advantage as $\mathcal{A}$, contradicting the LWE assumption. □

– *Hybrid* $\mathcal{H}_4$. In this hybrid, we replace ct_y with an encryption of zero.

Claim. $\mathcal{H}_3 \approx_c \mathcal{H}_4$ under the LWE assumption.

Proof. Computational indistinguishability between $\mathcal{H}_4$ and $\mathcal{H}_3$ follows immediately from the semantic security of FHE, and hence from LWE. □

At this point, it suffices to note that hybrid $\mathcal{H}_4$ is distributed identically to the output of $\mathcal{S}$, which concludes the proof of security for Bob.

This concludes the proof of Theorem 5. ■

Construction without output-succinctness. We remark that the proof of Theorem 5 does not make use of the BNIVOLE hiding property when arguing security for Alice; only when arguing security for Bob. This is because we already have statistical hiding for Alice's input thanks to the function-hiding transformation from Figure 1. Alternatively, we could avoid using the statistical-hiding transformation and just rely on BNIVOLE security to hide Alice's matrices as output by EvalPK; however, this would make the scheme less modular in the following sense. If the output-succinctness property (Definition 5) is *not* required, the construction from Figure 2 can simply have Alice's public encoding consist of $(\mathbf{A}_{C_i})_{i=1}^m$ such that Bob can locally compute $(\mathbf{s}^\top \mathbf{A}_{C_i})_{i=1}^m$. Specifically, we can simply remove the use of BNIVOLE in Figure 2, without changing the proof of security.

6 Construction from $i\mathcal{O}$

In this section, we construct SMS from $i\mathcal{O}$ in conjunction with other assumptions. Our construction supports the computation of *batch* functions over a large batch of short inputs provided by Alice and a short input provided by Bob. Concretely, we assume that Alice has a large batch of inputs $X = (x_1, \ldots, x_L)$ and Bob has a small input y, such that $|x_i| \approx |y|$. Using our construction, Alice and Bob can compute $f(x_i, y)$ for all $i \in [L]$, and for any function $f \in P/\mathsf{poly}$ determined adaptively *at decoding time*. We obtain input-output succinctness with respect to the batch size L.

Compared to our LWE-based construction from Section 5, our $i\mathcal{O}$-based construction supports all circuits in P/poly and is function adaptive (cf. Definition 7), allowing Alice and Bob to agree on the function they wish to compute over the entire batch or even for each individual entry in the batch.

6.1 Preliminaries

In this section, we provide the necessary preliminaries related to the $i\mathcal{O}$-based construction.

Indistinguishability obfuscation. Indistinguishability obfuscation ($i\mathcal{O}$) [BGI+01] satisfies the property that the obfuscation of two "functionally equivalent" circuits C_0 and C_1 are computationally indistinguishable.

Definition 14 (Indistinguishability Obfuscation [BGI+01]). *An efficient uniform algorithm* iO *is said to be an indistinguishability obfuscator for a class of circuits* $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *if the following properties hold:*

Correctness. *For all* $\lambda \in \mathbb{N}$*, for all* $C \in \mathcal{C}_\lambda$*, and for every input* x *to* C*:*

$$\Pr\Big[\ \widetilde{C} \leftarrow \mathsf{iO}(1^\lambda, C) \quad : \quad \widetilde{C}(x) = C(x)\ \Big] = 1,$$

where the probability is over the randomness of iO*.*

Security. *For all efficient distinguishers* $\mathcal{D}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, for all pairs of circuits* $C_0, C_1 \in \mathcal{C}_\lambda$ *such that* $C_0(x) = C_1(x)$ *on all inputs* x*,*

$$\Big|\Pr\Big[\ \mathcal{D}(\widetilde{C}_0) \quad : \quad \widetilde{C}_0 \leftarrow \mathsf{iO}(1^\lambda, C_0)\ \Big] - \Pr\Big[\ \mathcal{D}(\widetilde{C}_1) \quad : \quad \widetilde{C}_1 \leftarrow \mathsf{iO}(1^\lambda, C_1)\ \Big]\Big| \le \mathsf{negl}(\lambda),$$

where the probability is over the randomness of iO *and* $\mathcal{D}$*.*

Somewhere statistically binding hashing. Here, we recall the definition of somewhere statistically binding (SSB) hashing [HW15].

Definition 15 (Somewhere Statistically Binding Hashing [HW15]). *Let* λ *be a security parameter,* s *be a block length and* $\Sigma = \{0,1\}^s$ *be the block alphabet, and* $m = m(s) \in \mathsf{poly}(\lambda)$ *be the output length of the hash. Let* $p = p(s) \in \mathsf{poly}(\lambda)$ *be the opening size. A somewhere statistically binding (SSB) hash with local opening consists of four efficient algorithms* $\mathsf{SSB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, 1^s, L, i) \to \mathsf{hk}$*. The randomized key generation algorithm takes as input the security parameter* λ*, a block length* s*, an input length* $L \le 2^\lambda$*, and an index* $i \in [L]$*. It outputs a public hashing key* hk*.*
- $\mathsf{Hash}(\mathsf{hk}, X) \to c_X$*. The deterministic hashing algorithm takes as input the hash key* hk *and an input* $X = (x_1, \ldots, x_L) \in \Sigma^L$*. It outputs the hash value* $c_X \in \{0,1\}^m$*.*
- $\mathsf{Open}(\mathsf{hk}, x_j, j) \to \pi$*. The (possibly randomized) opening algorithm takes as input the hash key* hk*, a value* $x_j \in \Sigma$*, and an index* $j \in [L]$*. It creates an opening* $\pi \in \{0,1\}^p$*.*
- $\mathsf{Verify}(\mathsf{hk}, c_X, u, j, \pi) \to 0/1$*. The deterministic verification algorithm takes as input the hash key* hk*, a hash output* $c_X \in \{0,1\}^m$*, a value* $u \in \Sigma$*, an index* $j \in [L]$*, and an opening* $\pi \in \{0,1\}^p$*. It outputs a* 0 *(reject) or* 1 *(accept).*

The above functionality must satisfy the following properties:

Correctness. *For any block length* s*, any input length* L*, any pair of indices* $i, j \in [L]$*, and all* $X = (x_1, \ldots, x_L) \in \Sigma^L$*, it holds that:*

$$\Pr\left[\ \mathsf{Verify}(\mathsf{hk}, c_X, x_j, j, \pi) = 1 \quad : \quad \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, 1^s, L, i) \\ c_X := \mathsf{Hash}(\mathsf{hk}, X) \\ \pi \leftarrow \mathsf{Open}(\mathsf{hk}, x_j, j) \end{array}\right] = 1.$$

Index Hiding. *For all block lengths* s*, all input lengths* $L \le 2^\lambda$*, and all pairs of indices* $i_0, i_1 \in [L]$*,*

$$\Big\{\ \mathsf{hk} \ \Big|\ \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, 1^s, L, i_0)\ \Big\} \approx_c \Big\{\ \mathsf{hk} \ \Big|\ \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, 1^s, L, i_1)\ \Big\}.$$

Somewhere Statistically Binding. *The hash key* hk *is said to be* statistically binding with respect to the opening *for an index* $i \in [L]$ *if there do not exist any values* c_X*,* $x_i \ne x_i'$*, and openings* π, π' *such that* $\mathsf{Verify}(\mathsf{hk}, c_X, x_i, i, \pi) = \mathsf{Verify}(\mathsf{hk}, c_X, x_i', i, \pi') = 1$*. Formally, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all block lengths* s*, all input lengths* L*, and any index* $i \in [L]$*,*

$$\Pr\left[\ \begin{array}{c} \exists x_i, x_i' \in \Sigma \text{ such that } x_i \ne x_i' \\ \wedge \\ \mathsf{Verify}(\mathsf{hk}, c_X, x_i, i, \pi) = 1 \\ \wedge \\ \mathsf{Verify}(\mathsf{hk}, c_X, x_i', i, \pi') = 1 \end{array} \quad : \quad \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, 1^s, L, i)\ \right] = 1 - \mathsf{negl}(\lambda).$$

The hash function is said to be perfectly binding with respect to the opening *if the above probability is zero.*

Remark 9 (On perfect binding). Any perfectly-correct, rate-1 oblivious transfer (OT) scheme implies an SSB hash function with perfect binding via the transformation given in the work of Kalai et al. [KLVW23]. Moreover, the hash key is guaranteed to be indistinguishable from random if the OT receiver's message in the OT scheme is pseudorandom (which is indeed the case for constructions based on QR/DCR). We note that the construction of Hubáček and Wichs [HW15] when instantiated with a perfectly-correct FHE scheme (based on LWE) [BGV12] also gives an SSB hash construction with perfect binding with respect to the opening.

Theorem 6 ([HW15, KLVW23])**.** *Under either the QR, DCR, or the LWE assumption, there exists a construction of SSB hashing that has perfect binding with respect to the opening.*

Puncturable PRFs. We recall the notion of puncturable PRFs (PPRFs).

Definition 16 (Puncturable Pseudorandom Function [BW13, KPTZ13, BGI14])**.** *Let λ be a security parameter, $\mathcal{X} = \mathcal{X}_\lambda$ be the domain, and $\mathcal{Y}$ be the range. A puncturable pseudorandom function (PPRF) consists of three efficient algorithms* $\mathsf{PPRF} = (\mathsf{KeyGen}, \mathsf{Puncture}, \mathsf{Eval})$ *with the following syntax:*

- $\mathsf{KeyGen}(1^\lambda) \to K$*. The randomized key generation algorithm takes as input the security parameter λ and outputs a master key K.*
- $\mathsf{Puncture}(K, x^*) \to K^*$*. The randomized puncture algorithm takes as input a master key K and an input $x^* \in \mathcal{X}$. It outputs a punctured key K^*.*
- $\mathsf{Eval}(K, x) \to y$*. The deterministic evaluation algorithm takes as input a key K (which may be the punctured key) and an input $x \in \mathcal{X}$. It outputs a value $y \in \mathcal{Y}$.*

The above functionality must satisfy the following properties:

Correctness. *For all $\lambda \in \mathbb{N}$, every choice of punctured input $x^* \in \mathcal{X}$, and every $x \in \mathcal{X} \setminus \{x^*\}$, it holds that:*

$$\Pr\left[\mathsf{PPRF.Eval}(K, x) = \mathsf{PPRF.Eval}(K^*, x) \quad : \quad \begin{matrix} K \leftarrow \mathsf{KeyGen}(1^\lambda) \\ K^* \leftarrow \mathsf{Puncture}(K, x^*) \end{matrix}\right] = 1.$$

Security. *A puncturable PRF is a said to be* selective-puncturing secure *if for all efficient adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following security experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},b}(\lambda)$ is negligible in λ. Here, b denotes the challenge bit.*

1. *The challenger runs $\mathcal{A}(1^\lambda)$.*
2. *The adversary $\mathcal{A}$ sends a challenge $x^* \in \mathcal{X}$ to the challenger.*
3. *The challenger samples a master key $K \leftarrow \mathsf{KeyGen}(1^\lambda)$, and computes the punctured key $K^* \leftarrow \mathsf{Puncture}(K, x^*)$. Then, the challenger does the following:*
   - *If $b = 0$, compute $y := \mathsf{PPRF.Eval}(K, x^*)$ and respond with (K^*, y).*
   - *If $b = 1$, sample $y \leftarrow\!\!\$\, \mathcal{Y}$ and respond with (K^*, y).*
4. *$\mathcal{A}$ outputs a guess b', which is the output of the experiment.*

$\mathcal{A}$ wins if $b' = b$, and its advantage $\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda)$ is defined as

$$\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},1}(\lambda) = 1]\right|,$$

where the probability is over the randomness of $\mathcal{A}$ and KeyGen *and* $\mathsf{Puncture}$*.*

Remark 10 (t-puncturable PRF). We will also use the notion of a t-puncturable PRF [BCG+19], which can be realized in a black-box way using a 1-puncturable PPRF. A t-puncturable PRF is defined exactly as in Definition 16, except that the adversary is given a key punctured on t inputs and obtains t (real-or-random) challenges from the challenger. A simple and black-box construction of a t-puncturable PRF involves running t independent instances of a 1-puncturable PRF and defining the output to be the bit-wise XOR of all instances [BCG+19].

Theorem 7 (Existence of PPRFs [BW13,KPTZ13,BGI14])**.** *Assuming the existence of sub-exponentially-secure one-way functions, there exits a sub-exponentially-secure puncturable PRF (resp. t-puncturable PRF) with selective puncturing security.*

Commitment scheme. We require any commitment scheme with perfect binding. Such commitment schemes are known from injective one-way functions [BOV03].

Definition 17 (Perfectly-Binding Commitment Scheme)**.** *Let λ be a security parameter and $\mathcal{M}$ be a commitment message space. A perfectly-binding commitment scheme consists of an efficient algorithm* $\mathsf{Commit}(x;r) \to \widehat{x}$ *that takes as input a message $x \in \mathcal{M}$ and randomness $r \in \{0,1\}^\lambda$, and outputs a commitment $\widehat{x}$.* Commit *must satisfy the following properties:*

Perfect Binding. *For all $x, x' \in \mathcal{M}$ and every $r, r' \in \{0,1\}^\lambda$,*

$$\Pr\Big[\; x \neq x' \quad : \quad \mathsf{Commit}(x;r) = \mathsf{Commit}(x';r') \;\Big] = 0.$$

Computational Hiding. *For all efficient adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that:*

$$\Pr\left[\; \mathcal{A}(\widehat{x}_b, \mathsf{st}) \quad : \quad \begin{array}{r} (x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda) \\ r \leftarrow\!\!\$\, \{0,1\}^\lambda \\ b \leftarrow\!\!\$\, \{0,1\} \\ \widehat{x}_b := \mathsf{Commit}(x_b; r) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

6.2 Construction

We present our construction of SMS from $i\mathcal{O}$ in Figure 5. Our construction follows closely the ideas presented in Section 2.2 and makes use of an SSB hash function with perfect binding and $i\mathcal{O}$. Because the SSB hashing is performed over the set of *commitments* to Alice's inputs, we set the SSB hash input block length (denoted s) to be $\mathsf{poly}(\lambda, l)$ to accommodate the size of the commitment.

Remark 11 (On the use of the puncturable PRF). We note that the construction itself does not make use of the puncturable PRF, and instead treats it as a regular PRF. The requirement for a puncturable PRF appears only in the proof of security (see Appendix C.1).

The obfuscated program. Bob's obfuscated program is defined in Program 1 and is parameterized by a universal circuit $\mathcal{U} \in P/\mathsf{poly}$ that takes as input the tuple (f, x_i, y), consisting of a function description f, an input x_i, and an input y. The circuit $\mathcal{U}$ outputs $f(x_i, y)$. We assume that the size of $\mathcal{U}$ is polynomial in the security parameter λ, Alice's input length l and Bob's input length ℓ.

Program 1: (Parameterized by a universal circuit $\mathcal{U}$)

Hardcoded: (hk, K, y).
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**
2: $\quad R_i := \mathsf{PPRF.Eval}(K, c_{\widehat{X}} \| f \| i)$
3: $\quad d := \mathcal{U}(f, x_i, y)$
4: $\quad$ **return** $d \oplus R_i$
5: **else return** $\perp$

Fig. 6. Program obfuscated by Bob in Figure 5.

SMS from $i\mathcal{O}$

Public Parameters. We let L denote the size of Alice's batch of inputs $X = (x_1, \ldots, x_L)$, l denote the size of each input in the batch, such that $x_i \in \{0,1\}^l$ for all $i \in [L]$, ℓ denote the size of Bob's input y, Commit be a commitment scheme with perfect binding as defined in Definition 17, $\mathsf{SSB} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ be an SSB hash function (cf. Definition 15), and $\mathsf{PPRF} = (\mathsf{KeyGen}, \mathsf{Puncture}, \mathsf{Eval})$ be a puncturable PRF.

$\mathsf{SMS.Setup}(1^\lambda)$:

1 : $s := \mathsf{poly}(\lambda, l)$
2 : $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, 1^s, L, 0)$
3 : **return** $\mathsf{crs} := \mathsf{hk}$

$\mathsf{SMS.Encode}_A(\mathsf{crs}, 1^{|\mathcal{F}|}, X)$:

1 : **parse** $\mathsf{crs} = \mathsf{hk}$
2 : **parse** $X = (x_1, \ldots, x_L)$
3 : **foreach** $i \in [L]$:
4 : $\quad r_i \leftarrow\$ \{0,1\}^\lambda$
5 : $\widehat{x}_i := \mathsf{Commit}(x_i; r_i)$
6 : $\widehat{X} := (\widehat{x}_1, \ldots, \widehat{x}_L)$
7 : $c_{\widehat{X}} := \mathsf{SSB.Hash}(\mathsf{hk}, \widehat{X})$
8 : $\mathsf{pe}_A := c_{\widehat{X}}$
9 : $\mathsf{st}_A := (X, \widehat{X}, c_{\widehat{X}}, r_1, \ldots, r_L)$
10 : **return** $(\mathsf{pe}_A, \mathsf{st}_A)$

$\mathsf{SMS.Encode}_B(\mathsf{crs}, 1^{|\mathcal{F}|}, y)$:

1 : **parse** $\mathsf{crs} = \mathsf{hk}$
2 : $K \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$
3 : $\widetilde{P} \leftarrow \mathsf{iO}(1^\lambda, P)$, where P has (hk, K, y) hardcoded in it.
4 : $\mathsf{pe}_B := \widetilde{P}$
5 : $\mathsf{st}_B := K$
6 : **return** $(\mathsf{pe}_B, \mathsf{st}_B)$

$\mathsf{SMS.Decode}_A(\mathsf{crs}, f, \mathsf{pe}_B, \mathsf{st}_A)$:

1 : **parse** $\mathsf{crs} = \mathsf{hk}$ **and** $\mathsf{pe}_B = \widetilde{P}$
2 : **parse** $\mathsf{st}_A = (X, \widehat{X}, c_{\widehat{X}}, r_1, \ldots, r_L)$
3 : **parse** $X = (x_1, \ldots, x_L)$ **and** $\widehat{X} = (\widehat{x}_1, \ldots, \widehat{x}_L)$
4 : **foreach** $i \in [L]$:
5 : $\quad \pi_i := \mathsf{Open}(\mathsf{hk}, \widehat{x}_i, i)$
6 : $\quad z_A^{(i)} := \widetilde{P}(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$
7 : **return** $\mathbf{z}_A := (z_A^{(1)}, \ldots, z_A^{(L)})$

$\mathsf{SMS.Decode}_B(\mathsf{crs}, f, \mathsf{pe}_A, \mathsf{st}_B)$:

1 : **parse** $\mathsf{crs} = \mathsf{hk}$ **and** $\mathsf{pe}_A = c_{\widehat{X}}$ **and** $\mathsf{st}_B = K$
2 : **foreach** $i \in [L]$:
3 : $\quad z_B^{(i)} := \mathsf{PPRF.Eval}(K, c_{\widehat{X}} \| f \| i)$
4 : **return** $\mathbf{z}_B := (z_B^{(1)}, \ldots, z_B^{(L)})$

Fig. 5. Simultaneous-Message and Succinct Secure Computation from $i\mathcal{O}$.

6.3 Setting the parameters

Here, we explain how we need to set the parameters for the underlying primitives to achieve security for our full construction by complexity leveraging.

Let λ be the security parameter for the $i\mathcal{O}$-based SMS construction. We let the security parameter of the SSB hash, denoted λ_{ssb}, be the same as λ. Let $\lambda_{\mathsf{io}} = q(\lambda)$ and $\lambda_{\mathsf{pprf}} = q(\lambda)$ be polynomial in the security parameter λ, for some polynomial $q(\lambda) \in \mathsf{poly}(\lambda)$, which we will set later.

Next, we let ϵ_{io}, ϵ_{pprf}, and ϵ_{ssb}, denote the advantage of any efficient distinguisher $\mathcal{D}$ in the $i\mathcal{O}$ security game (cf. Definition 14), the PPRF security game (cf. Definition 16), and the SSB index-hiding game (cf. Definition 15), respectively. Furthermore, let the PPRF domain length be $n = \mathsf{poly}(\lambda, \log L)$, where L is Alice's batch length. Define $\epsilon \geq \max(\epsilon_{\mathsf{io}}, \epsilon_{\mathsf{pprf}})$. Then, we have that $1/2^{q^\epsilon}$ bounds the advantage of the $\mathcal{D}$ in the $i\mathcal{O}$ game and the PPRF game.

Now, we need to set q such that $2^n/2^{q^\epsilon}$ is negligible in λ. This can be achieved by choosing q such that $q^\epsilon \geq O(n + \lambda)$, which remains polynomial in the security parameter.

6.4 Security analysis

In this section, we analyze the correctness and security of our $i\mathcal{O}$-based construction of SMS from Figure 5. We prove the following theorem:

Theorem 8. *Assuming the existence of sub-exponentially-secure indistinguishability obfuscation and the existence of sub-exponentially-secure one-way functions, in addition to the existence of somewhere statistically binding hash functions (with perfect binding) and injective one-way function, Figure 5 is a $O(\log\log L/\log L)$-succinct SMS scheme supporting all batched function families in P/poly, where L is the batch size.*

In Proposition 1 we prove the correctness and succinctness of the construction. In Proposition 2, we prove security for Alice. Then, in Proposition 2, we prove security for Bob.

Proposition 1. *Figure 5 satisfies the correctness properties of Definition 4 and $O(\log\log L/\log L)$-batch-succinctness (cf. Definition 9).*

Proof. Consider Program 1. For each $i \in [L]$, notice that if π_i is a valid opening for $\widehat{x}_i$ with respect to the SSB hash value $c_{\widehat{X}}$ and index i, and (x_i, r_i) is a valid decommitment for $\widehat{x}_i$, then the output of the obfuscated program is $z_A^{(i)} := f(x_i, y) \oplus \mathsf{PPRF.Eval}(K, c_{\widehat{X}} \| f \| i)$. Bob's output, on the other hand, is $z_B^{(i)} := \mathsf{PPRF.Eval}(K, c_{\widehat{X}} \| f \| i)$. The parties thus obtain additive shares of $f(x_i, y)$, by the correctness of $i\mathcal{O}$. Furthermore, because this holds for all $i \in [L]$, the output of Decode_σ is an additive share of the function applied to components of Alice's batch of inputs X, as required.

Finally, we note that the size of the obfuscated circuit depends polynomially on the input and output lengths ℓ, l, m, but only depends *logarithmically* on the batch size L (the dependence on L comes from domain of the PRF needing to be large enough to accommodate the index i of the batch input). Hence, we have that the size of the program obfuscated by Bob is of size $\mathsf{poly}(\lambda, \ell, l, m, \log L)$, which gives $\epsilon = O(\log\log L/\log L)$ batch-succinctness since $L^{O(\log\log L/\log L)} = \mathsf{polylog}(L)$. ■

Security for Alice. Proving security for Alice is relatively straightforward and comes down to the computational-hiding property of the commitment scheme. We prove the following proposition.

Proposition 2. *Figure 5 satisfies the security property of Definition 4 for Alice assuming the commitment scheme* Commit *is computationally hiding.*

Proof. Consider the following efficient simulator $\mathcal{S}$ for pe_A.

$\mathcal{S}$: On input crs,
- Parse $\mathsf{crs} = \mathsf{hk}$.
- Sample $r_1, \ldots, r_L \leftarrow\$ \{0,1\}^\lambda$.
- Set $\widehat{x}_i := \mathsf{Commit}(0; r_i)$.
- $c_{\widehat{X}} := \mathsf{SSB.Hash}(\mathsf{hk}, (\widehat{x}_1, \ldots, \widehat{x}_L))$.
- Output $\mathsf{pe}_A := c_{\widehat{X}}$.

We now argue that the output of $\mathcal{S}$ is computationally indistinguishable to the output of Encode_A. Suppose, towards contradiction, that there exists an efficient distinguisher $\mathcal{A}$ that distinguishes, with non-negligible advantage, between the view produced by $\mathcal{S}$ and pe_A produced by Encode_A in Figure 5. Notice that the only difference in the simulated view compared to Encode_A is that each $\widehat{x}_i$ is a commitment to zero. As such, by a straightforward hybrid argument, $\mathcal{A}$ breaks the computational hiding of Commit, which raises a contradiction. This concludes the proof. ■

Security for Bob. Proving security for Bob is more involved and requires carefully removing the presence of Bob's input y from the obfuscated program via a sequence of hybrids. More concretely, our proof strategy involves iterating over all possible inputs $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$ to Bob's program and carefully "puncturing" the program at the j-th canonical input to the PRF by programming the output to be a uniformly random string that is independent of the input y. In the process, we make use of the SSB hash to guarantee functional equivalence and index hiding, which then allows us to invoke $i\mathcal{O}$ security.

This overall strategy requires us to consider an *exponential* (in the SSB hash security parameter) number of hybrids, which requires complexity leveraging and assuming sub-exponential security of the underlying primitives (namely, sub-exponentially secure $i\mathcal{O}$ and one-way functions). See Section 6.3 for how we set parameters to obtain sub-exponential security.

Proposition 3. *Figure 5 satisfies the security property of Definition 4 for Bob assuming the security of the SSB hash (cf. Definition 15), the existence of injective one-way functions, the existence of sub-exponentially secure indistinguishability obfuscation (cf. Definition 14), and the existence of sub-exponentially secure one-way functions.*

Proof. Deferred to Appendix C.1. ■

7 Optimizations

In this section, we discuss optimizations that we can make to our LWE-based and $i\mathcal{O}$-based constructions that further push the communication and computational efficiency.

7.1 Unbounded computations

We show that our LWE-based construction can be upgraded to support *unbounded* computations assuming the circular security of LWE. In particular, the recent result of Hsieh, Lin, and Luo [HLL23] show how to construct algorithms EvalPK and EvalCT supporting unbounded computations while satisfying the properties of Lemma 3. In particular, we can use the following lemma (where the highlighted parts indicate the difference with Lemma 3).

Lemma 5 (Adapted from [HLL23, Theorem 13] and [GVW15, Lemma 3.2])**.** *Let $\lambda \in \mathbb{N}$ be a security parameter. Under the circular security of the $\mathsf{LWE}_{n,k,q,\tau}$ assumption (as defined in [HLL23, Assumption 1]) with $k := n\lceil \log q \rceil$, there exist algorithms $(\mathsf{EvalPK}, \mathsf{EvalCT})$ satisfying Definition 13 for all integers $\alpha = \alpha(\lambda), \beta = \beta(\lambda)$ that are polynomial in the security parameter, such that for all common random strings of the form: $\mathsf{crs} := (\mathbf{A}_1, \ldots, \mathbf{A}_\alpha, \mathbf{B}_1, \ldots, \mathbf{B}_\beta) \in (\mathbb{Z}_q^{n \times k})^{\alpha+\beta}$, for all $\alpha + \beta$ vectors $\mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta \in \mathbb{Z}_q^k$, all $\mathbf{s} \in \mathbb{Z}_q^n$, all $(x, \mathbf{y}) \in \{0,1\}^\alpha \times \mathbb{Z}_q^\beta$, and all arithmetic circuits $C \colon \{0,1\}^\alpha \to \mathbb{Z}_q^\beta$ in P/poly, if it holds that:*

$$\forall i \in [\alpha],\ \mathbf{u}_i = \mathbf{s}^\top(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i^\top \text{ and } ||\mathbf{e}_i||_\infty \leq B,$$
$$\forall i \in [\beta],\ \mathbf{v}_i = \mathbf{s}^\top(\mathbf{B}_i + \mathbf{y}[i] \cdot \mathbf{G}) + \mathbf{e}_i^\top \text{ and } ||\mathbf{e}_i||_\infty \leq B,$$

then it also holds that for $\mathbf{w}_{\mathsf{IP}\circ C} := \mathsf{EvalCT}(\mathsf{crs}, \mathbf{u}_1, \ldots, \mathbf{u}_\alpha, \mathbf{v}_1, \ldots, \mathbf{v}_\beta, C, x)$,

$$\mathbf{w}_{\mathsf{IP}\circ C} = \mathbf{s}^\top(\mathbf{A}_{\mathsf{IP}\circ C} + \langle C(x), \mathbf{y} \rangle \cdot \mathbf{G}) + \mathbf{e}^\top \text{ with } ||\mathbf{e}||_\infty \leq B,$$

where $\mathbf{A}_{\mathsf{IP}\circ C} := \mathsf{EvalPK}(\mathsf{crs}, C)$ and $\mathbf{G}$ is the gadget matrix from Definition 12.

Proof (sketch). The lemma follows from [HLL23, Theorem 13] coupled with [GVW15, Lemma 3.2]. [HLL23, Theorem 13] proves the existence of EvalPK and EvalCT as defined in Definition 13 (without the extension to the class $\mathsf{IP} \circ C$) while [GVW15, Lemma 3.2] provides a generic extension to the class $\mathsf{IP} \circ C$. In particular, the proof of [GVW15, Lemma 3.2] follows directly from the properties satisfied by these two algorithms and therefore also applies to the (unbounded) variants. ■

As an immediate corollary of Lemma 5, we have that the LWE-based construction of SMS works for any *unbounded* depth function (with a polynomially-sized circuit representation) assuming the circular security of LWE. In particular, we note that Lemma 4 (the function-hiding transformation), introduces a fixed additive factor overhead to the error and is not impacted by the underlying instantiation of EvalPK and EvalCT. As such, we have all the ingredients to instantiate Figure 2 for unbounded-depth computations.

7.2 Minimizing communication from Bob to Charlie

We observe that in our $i\mathcal{O}$-based construction (cf. Figure 5), Bob's output consists solely of a pseudorandom string computed by the PRF evaluated on input $(c_{\widehat{X}} \| f \| i)$, for i ranging from 1 to L.

At first glance, it may appear that Bob can simply send the PRF key K to Charlie, making the communication overhead from Bob to Charlie optimal (similarly to the communication overhead in the naive FHE-based example described in Section 1). However, this idea is obviously insecure in the

case where Charlie colludes with Alice, since it would allow Alice to recover the PRF value on *all* inputs to the obfuscated program and perform a resetting attack.

Nonetheless, we show that we can easily tweak this idea by resorting to a *constrained* PRF (CPRF), which restricts the domain of the PRF that Charlie is allowed to evaluate. Specifically, a CPRF generalizes the notion of a puncturable PRF and can be constrained on an arbitrary predicate (not just the puncturing "index" predicate). We provide a formal definition of CPRFs in Appendix B.

Our idea is to have Bob constrain the PRF on all inputs outside of the set $S = \{(c_{\widehat{X}} \| f) \| i \mid i \in [L]\}$. In particular, given a constrained key constrained to the set S, only L values of the PRF can be evaluated, and these values correspond exactly to the output of Bob. However, because the predicate has a succinct description of size $O(\log L)$, the communication from Bob to Charlie is only polylogarithmic in the batch size L (ignoring polynomial factors in the security parameter).

Connection to Hubáček–Wichs. We additionally observe that when the communication from Bob to Charlie is made succinct, the resulting SMS scheme immediately implies a two-round secure computation protocol that is succinct in the output length. This shares a close connection with the protocol of Hubáček and Wichs [HW15] (and thus inherits the same impossibility results discussed therein related to secure computation protocols where the total communication is succinct in the output length). Specifically, in an SMS scheme, we can assume that Alice and Charlie are colluding. Therefore, Bob can simply send the succinct output message to Alice, who can then locally recover the output. This protocol is two rounds and is similar in flavor to the "multi-decryption" protocol of Hubáček and Wichs [HW15, Section 3.2] (indeed, our construction generalizes their notion to any batch computation).

Programming the output. Interestingly, to prove our optimized construction secure, we need to resort to the same techniques used by Hubáček and Wichs [HW15]. That is, in order to simulate the view of Alice, the simulator needs to program the output (which is now much longer than the view of Bob) and therefore, the simulator has no choice but to program the output into the random coins of Alice. This makes the construction only possible in the honest-but-curious model, where the simulator can specify the randomness used by Alice.[11] Hubáček and Wichs [HW15] provide several impossibility results ruling out the alternative approaches, and even a weaker "honest-but-deterministic" adversarial model.

Compressed decoding and construction. In Definition 18, we formalize the notion of a compressed decoding procedure for Bob, which we will instantiate by adapting the $i\mathcal{O}$-based construction from Figure 5 in Figure 7.

Definition 18 (Compressed Decoding)**.** *We say an SMS scheme supports a compressed decoding procedure for Bob if the algorithm* Decode_B *can be decomposed into two efficient algorithms* $(\mathsf{DecodeCompressed}, \mathsf{DecodeExpand})$ *with the following syntax:*

- $\mathsf{DecodeCompressed}(\mathsf{crs}, f, \mathsf{pe}_A, \mathsf{st}_B) \to \tilde{z}_B$*. The deterministic compressed decoding algorithm takes as input the CRS* crs*, a function* $f \in \mathcal{F}$*, the public encoding* pe_A *belonging to Alice, and secret state* st_B *belonging to Bob. It outputs a compressed string* $\tilde{z}_B$*.*
- $\mathsf{DecodeExpand}(\tilde{z}_B) \to z_B$*. The deterministic expansion algorithm takes as input the string* $\tilde{z}_B$ *output by* $\mathsf{DecodeCompressed}$*. It outputs an* m*-bit string* $z_B \in \{0,1\}^m$*.*

The above algorithms must satisfy the following correctness, succinctness, and security properties.

Correctness. *For all inputs* input *in the domain of* Decode_B*, it holds that:*

$$\Pr\Big[\mathsf{Decode}_B(\mathsf{input}) = \mathsf{DecodeExpand}(\mathsf{DecodeCompressed}(\mathsf{input}))\Big] = 1.$$

Compactness. *There exists an* $\epsilon \in [0,1)$ *such that for all security parameters* $\lambda \in \mathbb{N}$*, every input* input *in the domain of* Decode_B*, it holds that:*

$$|\mathsf{DecodeCompressed}(\mathsf{input})| \leq \mathsf{poly}(\lambda) \cdot |\mathsf{Decode}(\mathsf{input})|^{\epsilon}.$$

[11] We note that this model can be upgraded to a malicious setting in the random oracle model, since the random oracle can be programmed to produce the correct random coins needed for the simulation.

Security. *Let* $\mathsf{poly}(\cdot)$ *be a fixed polynomial. There exists an efficient simulator* $\mathcal{S}$ *such that for all* crs *in the support* Setup*, for all* $f \in \mathcal{F}$*, and all inputs* X, y*,*

$$\left\{ (\mathsf{coins}, \mathsf{pe}_B, \tilde{z}_B) \;\middle|\; \begin{array}{r} \mathsf{coins} \leftarrow\!\$\, \{0,1\}^{\mathsf{poly}(\lambda, |X|)} \\ (\mathsf{pe}_A, \mathsf{st}_A) \leftarrow \mathsf{Encode}_A(\mathsf{crs}, f, X; \mathsf{coins}) \\ (\mathsf{pe}_B, \mathsf{st}_B) \leftarrow \mathsf{Encode}_B(\mathsf{crs}, f, y) \\ \tilde{z}_B := \mathsf{DecodeCompressed}(\mathsf{crs}, f, \mathsf{pe}_A, \mathsf{st}_B) \end{array} \right\} \approx_c \mathcal{S}(\mathsf{crs}, f, f(X,y), X).$$

Compressed Decoding

Parameters. Let $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ be a constrained PRF (cf. Definition 23).

Changes to Encode_A and Encode_B in Figure 5.

- Encode_A "augments" $X = (x_1, \ldots, x_L)$ to consist of L tuples $X^{\mathsf{aug}} := \big((x_1, r_1), \ldots, (x_L, r_L)\big)$, where $r_1, \ldots, r_L \leftarrow\!\$\, \{0,1\}^m$ are random masks of the same length as the output length m.
- Encode_B obfuscates an augmented program described in Program 2, which ignores the randomness component of each input tuple but is otherwise identical to Program 1.

$\mathsf{SMS.DecodeCompressed}_B(\mathsf{crs}, f, \mathsf{pe}_A, \mathsf{st}_B)$:

1 : **parse** $\mathsf{crs} = \mathsf{hk}$ **and** $\mathsf{pe}_A = c_{\widehat{X}}$ **and** $\mathsf{st}_B = K$
2 : $S := \{c_{\widehat{X}} \| f \| i \mid i \in [L]\}$
3 : Define the predicate P: $x \mapsto x \in S$
4 : $\mathsf{csk} \leftarrow \mathsf{CPRF.Constrain}(K, P)$
5 : $\tilde{z}_B := (c_{\widehat{X}}, f, \mathsf{csk})$
6 : **return** $\tilde{z}_B$

$\mathsf{SMS.DecodeExpand}(\tilde{z}_B)$:

1 : **parse** $\tilde{z}_B = (c_{\widehat{X}}, f, \mathsf{csk})$
2 : **foreach** $i \in [L]$:
3 : $\quad z_B^{(i)} := \mathsf{CPRF.CEval}(\mathsf{csk}, c_{\widehat{X}} \| f \| i)$
4 : **return** $\mathbf{z}_B := (z_B^{(1)}, \ldots, z_B^{(L)})$

Fig. 7. Compressed decoding procedure for the $i\mathcal{O}$-based SMS scheme from Figure 5.

Program 2: (Parameterized by a universal circuit $\mathcal{U}$)

Hardcoded: (hk, K, y).
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**
2: $\quad$ **parse** $x_i = (x_i', \_)$ // Parse x_i as a tuple and ignore the second component.
3: $\quad R_i := \mathsf{PPRF.Eval}(K, c_{\widehat{X}} \| f \| i)$
4: $\quad d := \mathcal{U}(f, x_i', y)$
5: $\quad$ **return** $d \oplus R_i$
6: **else return** $\perp$

Fig. 8. Program obfuscated by Bob in Figure 7.

Proposition 4. *The compressed decoding procedure from Figure 7 satisfies Definition 18 with respect to the $i\mathcal{O}$-based SMS scheme from Figure 5.*

Proof. We prove each property in turn. We note, in passing, that CPRFs are known for all constraint predicates assuming $i\mathcal{O}$ and one-way functions [BZ14, BLW17].

Correctness. We first note that Program 1 and Program 2 are functionally equivalent and so the output computed by Alice in Decode_A is unchanged. Correctness then follows immediately from the

correctness of the CPRF (cf. Definition 23). In particular, the CPRF correctness guarantees that the evaluation on all inputs in the set S matches the evaluation of the CPRF on under the master key K. Therefore, when considering the subset of the domain on which the PRF is evaluated in Decode_B as defined in Figure 5, the evaluation using the constrained key csk is guaranteed to be identical.

Compactness. The CPRF already guarantees that $|\mathsf{csk}| \in \mathsf{poly}(\lambda, |P|)$, where P is predicate. Then, because our predicate is a range predicate over the domain $[L]$, it has size $O(\log L)$. We therefore have that $|\mathsf{csk}| = m^{\epsilon} \cdot \mathsf{poly}(\lambda)$ with $\epsilon = \log\log L / \log L$, where m is the output length.

Security. We first describe the simulator $\mathcal{S}$.

$\mathcal{S}$: On input $(\mathsf{crs},\ f,\ f(X,y), X)$:

- Parse $\mathsf{crs} = \mathsf{hk}$, $f(X,y) = \big(f(x_1,y),\ldots,f(x_L,y)\big)$, $X = (x_1,\ldots,x_L)$.
- Sample $\mathsf{rnd} \leftarrow\!\$\ (\{0,1\}^{\lambda})^L$ and $r_1,\ldots,r_L \leftarrow\!\$\ \{0,1\}^m$.
- Set $X^{\mathsf{aug}} := \big((x_1, r_1 \oplus f(x_1,y)),\ldots,(x_L, r_L \oplus f(x_L,y))\big)$.
- $(\mathsf{pe}_A, \mathsf{st}_A) := \mathsf{Encode}_A(\mathsf{crs}, 1^{|\mathcal{F}|}, X^{\mathsf{aug}}; \mathsf{rnd})$. // Run Encode_A from Figure 5 with coins rnd.
- Parse $\mathsf{pe}_A = c_{\widehat{X}}$.
- Sample $K \leftarrow \mathsf{CPRF.KeyGen}(1^{\lambda})$ and $K_0 \leftarrow \mathsf{CPRF.KeyGen}(1^{\lambda})$.
- Compute $\widetilde{P^{\mathsf{sim}}} \leftarrow \mathsf{iO}(1^{\lambda}, P^{\mathsf{sim}})$, where P^{sim} is as described in Program 3 and has hardcoded inputs $(\mathsf{hk}, K_0, \mathsf{csk}, c_{\widehat{X}}, f)$.
- Define the set $S := \big\{c_{\widehat{X}} \| f \| i \mid i \in [L]\big\}$.
- Compute $\mathsf{csk} \leftarrow \mathsf{CPRF.Constrain}(K, S)$.
- Set $\mathsf{pe}_B := \widetilde{P^{\mathsf{sim}}}$ and $\tilde{z}_B := (c_{\widehat{X}}, f, \mathsf{csk})$.
- Set $\mathsf{coins} := (\mathsf{rnd}, r_1 \oplus f(x_1,y),\ldots,r_L \oplus f(x_L,y))$.
- Output $(\mathsf{coins}, \mathsf{pe}_B, \tilde{z}_B)$.

Program 3: The Simulated Program

Hardcoded: $(\mathsf{hk}, K_0, \mathsf{csk}, c_{\widehat{X}}, f)$.
Input: $(c'_{\widehat{X}}, (x_i, r_i), (\widehat{x}_i, r'_i), i, \pi_i, f')$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}((x_i, r_i); r'_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c'_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**

if $c'_{\widehat{X}} = c_{\widehat{X}} \wedge f' = f$ **then**	**else**
1: $R_i := \mathsf{PPRF.CEval}(\mathsf{csk}, c_{\widehat{X}} \| f \| i)$	1: $R_i := \mathsf{PPRF.Eval}(K_0, c'_{\widehat{X}} \| f' \| i)$
2: **return** $R_i \oplus r_i$	2: **return** $R_i \oplus r_i$

2: **else return** $\perp$

We now prove security via a hybrid argument.

- *Hybrid* $\mathcal{H}_0$. This hybrid consists of $(\mathsf{crs}, \mathsf{coins}, \mathsf{pe}_B, \tilde{z}_B)$, where pe_B is an obfuscation of Program 1.
- *Hybrid* $\mathcal{H}_1$. In this hybrid, we set coins to be as computed by $\mathcal{S}$.

  Claim. $\mathcal{H}_1 \approx_s \mathcal{H}_0$.

  Proof. $\mathcal{H}_1$ is perfectly indistinguishable from $\mathcal{H}_0$ since $r_i \oplus f(x_i, y)$ is distributed identically to a uniformly random value when r_i is sampled uniformly. □

- *Hybrid* $\mathcal{H}_2$. In this hybrid, we replace pe_B with an obfuscation of Program 3.

  Claim. $\mathcal{H}_2 \approx_c \mathcal{H}_1$ under the same assumptions as required for Lemma 8.

  Proof (sketch). The proof follows as a corollary of the proof of Lemma 8. The sequence of hybrids in the proof of Proposition 3 are used to prove that a program that just outputs a PRF evaluation

on all inputs is computationally indistinguishable from a program that outputs a PRF evaluation under a key K_0 for all inputs smaller than the j-th canonical input and output a secret masked by a PRF evaluation under a key K on all other inputs.

Without loss of generality, we can reorder the hybrids such that all inputs to the PRF that are prefixed by $c_{\widehat{X}} \| f$ are evaluated under PRF key K and all other inputs are evaluated using the independent PRF key K_0. In doing so, we use the CPRF (rather than the puncturable PRF) to go from one hybrid to the next.

Eventually, K_0 is only used on inputs that are not prefixed by $c_{\widehat{X}} \| f$, and the key K is used on all other inputs. Then, because of the fact that csk is the constrained key derived from K, and the evaluation using csk is equivalent to the evaluation under K for all inputs prefixed by $c_{\widehat{X}} \| f$, we can replace K with csk while keeping the programs functionally equivalent. □

At this point, $\mathcal{H}_2$ is identical to the output of $\mathcal{S}$, which concludes the proof. ■

7.3 Minimizing computation for Bob

We show an additional optimization allowing us reduce the computational complexity for Bob in certain cases. In particular, we consider the case where the output of the batch function is summed together, using Remark 6 (post-composition with a linear function). In this case, while the intermediate result held by Alice and Bob is of length $l \cdot L$, the final output is just one block of length l. Having Bob compute the full intermediate shares in this scenario results in "wasteful" computation, since the final output computed by Bob is just a pseudorandom share of length l rather than $l \cdot L$. As we will show, Bob's computation can be reduced to just $\mathsf{poly}(\lambda, \log L)$ as opposed to $\mathsf{poly}(\lambda, L)$ by resorting to an aggregatable PRF [CGV15]. In a nutshell, an aggregatable PRF allows the party with the PRF key K to compute $\bigoplus_{i=a}^{b} F_K(i)$, for any range $[a, b]$ in the domain, in the same time it takes to evaluate the PRF F_K on a single input in the domain.

As was observed in Section 7.2, Bob's output consists of the PRF evaluated on input $(c_{\widehat{X}} \| f \| i)$, for i ranging from 1 to L. In particular, this share is computed by evaluating the PRF on a finite range of consecutive inputs in the domain, which are then summed together to derive the final share. Therefore, using an aggregatable PRF, the computation can be performed in $\mathsf{poly}(\lambda, \log L)$ time. However, because we require a *puncturable* PRF for the security proof, we must first construct what we call a puncturable aggregatable PRF (PAPRF) to be able to apply this optimization.

7.3.1 Puncturable Aggregatable PRF A PRF that is both puncturable and aggregatable allows aggregating the PRF over a range $[a, b]$ using the master key, while given the punctured key, it should not be possible to evaluate the PRF on the punctured input $c \in [a, b]$ (or aggregate over a range that includes the punctured input).

Before explaining how we construct a puncturable aggregatable PRF (PAPRF), we first recall the black-box construction of Cohen, Goldwasser, and Vaikuntanathan [CGV15] for aggregatable PRFs supporting summation over a finite range.[12] Let G be any PRF. The aggregatable PRF F is constructed as:

$$G_K(x) = \begin{cases} F_K(0) & : x = 0, \\ F_K(x) \oplus F_K(x-1) & : x \neq 0. \end{cases}$$

Note that the PRF F_K is efficiently aggregatable on any range $[a, b]$ in the domain:

$$\bigoplus_{x \in [a,b]} G_K(x) = F_K(b) \oplus F_K(a-1).$$

If we want to puncture the above PRF on some input $c \in [a, b]$, then we need to puncture F_K on *two* points in the range, namely c and $c + 1$, using a "t-puncturable" PRF (cf. Remark 10). In particular, if the punctured key only prevents evaluating the aggregatable PRF on input $c \in [a, b]$ but still enables aggregation elsewhere, then the punctured key can be used to recover $F_K(c)$, breaking

[12] We will not formally define the notion of aggregatable PRFs since we only focus on the simple case of summation over a range, which has a simple black-box construction.

puncturing security. To see this, note that it is possible to compute the aggregate over three ranges: $[a, c-1]$, $[c+1, b]$ and $[a, b]$ to then recover the PRF value on c by computing:

$$\bigoplus_{x\in[a,b]} F_K(x) \oplus \sum_{x\in[a,c-1]} F_K(x) \oplus \bigoplus_{x\in[c+1,b]} F_K(x) = F_K(c).$$

To get around this issue, we therefore require puncturing two consecutive inputs at a time.

We explain our construction next.

PAPRF construction. We present the black-box construction of PAPRF in Figure 9. Our construction simply instantiates the aggregatable PRF using a 2-puncturable PRF.

Puncturable Aggregatable PRF

Parameters. Let F be a 2-puncturable PRF constructed (cf. Remark 10).

$\mathsf{PAPRF.KeyGen}(1^\lambda)$:
1 : $K \leftarrow F.\mathsf{KeyGen}(1^\lambda)$
2 : **return** K

$\mathsf{PAPRF.Puncture}(K, x)$:
1 : $S := \{x, x+1\}$
2 : $K^* \leftarrow F.\mathsf{Puncture}(K, S)$
3 : **return** K^*

$\mathsf{PAPRF.Eval}(K, x)$:
1 : **if** $x = 0$: $y := F_K(x)$
2 : **else** $y := F_K(x) \oplus F_K(x-1)$
3 : **return** y

Fig. 9. Puncturable Aggregatable PRF.

Lemma 6. *The PAPRF construction in Figure 9 is puncturable on any pair of consecutive inputs in the domain.*

Proof. Suppose, towards contradiction, that there exists an efficient adversary $\mathcal{A}$ that wins the puncturable PRF security game (extended to the 2-puncturing case in the natural way) with non-negligible advantage $\nu(\lambda)$ against the PAPRF construction.

Now, consider the following sequence of hybrid games.

- *Hybrid* $\mathcal{H}_0$. This hybrid consists of the 2-puncturable PRF game.
- *Hybrid* $\mathcal{H}_1$. In this hybrid game, the 2-puncturing PRF challenger outputs a real-or-random challenge on one of the two punctured inputs (the other input is now always pseudorandom). It follows that $\mathcal{A}$'s advantage in $\mathcal{H}_1$ is at least $\nu/2$. Specifically, recall that we are considering F to be a 2-puncturable PRF as constructed in Remark 10 by taking the bit-wise XOR of the output of two independent 1-puncturable PRF evaluations.

We can now construct an efficient $\mathcal{B}$ breaking the 1-puncturing security of an underlying puncturable PRF instance used to realize the 2-puncturing PRF via Remark 10. $\mathcal{B}$ proceeds as follows:

1. Receive input x^* as the 1-punctured PRF challenge from $\mathcal{A}$.
2. Forward x^* to the challenger and receive the 1-punctured PRF key K_0^*.
3. Generate a fresh 1-puncturable PRF master key K_1 along with the corresponding punctured key K_1^*, punctured on input $x^* + 1$.
4. Define the 2-punctured key $K^* = (K_0^*, K_1^*)$.
5. Respond to $\mathcal{A}$ with K^*.
6. For each query x issued by $\mathcal{A}$:
   - Query the challenger on x to get response $y_0^{(x)}$ and compute $y_1^{(x)} := F_{K_1}(x)$.
   - Compute $y_0^{(x-1)} := F_{K_0^*}(x-1)$ and $y_1^{(x-1)} := F_{K_1}(x-1)$.
   - Respond with $(y_0^{(x)} \oplus y_1^{(x)}) \oplus (y_0^{(x-1)} \oplus y_1^{(x-1)})$.
7. Output as $\mathcal{A}$ does.

First, an admissible $\mathcal{A}$ never queries $\mathcal{B}$ on input $x^* + 1$, so $\mathcal{B}$ answers all queries correctly and the responses are distributed identically to either $\mathcal{H}_0$ or $\mathcal{H}_1$, depending on the challenge. Therefore, $\mathcal{B}$ has advantage $\nu/2$ in breaking the 1-puncturing security game against F, which contradicts the puncturing security of F. ■

Lemma 7. *The PAPRF construction in Figure 9 is aggregatable over interval subsets of the domain.*

Proof. The construction is the same as the one of Cohen et al. [CGV15]. It follows that:

$$\bigoplus_{x \in [a,b]} \mathsf{PAPRF.Eval}(K, x) = F_K(b) \oplus F_K(a-1).$$

■

Remark 12 (Using a PAPRF with Figure 5). By using a PAPRF instead of a PPRF in Figure 5, Bob's computation time in the case where Alice and Bob compute an aggregation over their intermediate shares is reduced to $\mathsf{poly}(\lambda, \log L)$. Moreover, the security proof from Proposition 3 remains nearly identical with the only exception being that the PPRF is replaced with a 2-puncturable PRF. This requires puncturing two consecutive inputs at a time in the relevant hybrids but does not otherwise change the proof and analysis.

8 Rate-1 FHE from SMS

In this section, we show that SMS can be used to compile any fully homomorphic encryption scheme into a rate-1 FHE scheme. We recall the definition of a rate-1 FHE [BDGM19].

At a high level, a rate-1 FHE scheme is an FHE scheme (cf. Definition 2) adorned with additional algorithms to compress a vector of ciphertexts into a compact representation that approaches the message length, asymptotically.

Definition 19 (Rate-1 Fully Homomorphic Encryption [BDGM19])**.** *A* rate-1 *FHE scheme consists of an FHE scheme* $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *adorned with additional algorithms* $(\mathsf{Compress}, \mathsf{CompressDec})$ *with the following syntax:*

- $\mathsf{Compress}(\mathsf{pk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_m)) \to \mathsf{ct}^*$. *The deterministic compression algorithm takes as input the public key* pk *and* m *(possibly evaluated) ciphertexts* $(\mathsf{ct}_1, \ldots, \mathsf{ct}_m)$*. It outputs a compressed ciphertext* ct^**.*
- $\mathsf{CompressDec}(\mathsf{sk}, \mathsf{ct}^*) \to x$*. The deterministic compressed decryption algorithm takes as input a compressed ciphertext, the secret key* sk*, and a compressed ciphertext* ct^**. It outputs the message* x*.*

The above algorithms must satisfy the following properties:

Compressed Correctness. *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all messages* $x_1, \ldots, x_\ell \in \mathcal{M}$ *and all* ℓ*-argument,* m*-output functions* f *that can be represented by polynomial-size circuits, we have that:*

$$\Pr\left[\begin{array}{rcl} \mathsf{CompressDec}(\mathsf{sk}, \mathsf{ct}^*) & : & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ = f(x_1, \ldots, x_\ell) & & \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i), \forall i \in [\ell] \\ & & \mathsf{ct}'_j \leftarrow \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)), \forall j \in [m] \\ & & \mathsf{ct}^* \leftarrow \mathsf{Compress}(\mathsf{pk}, (\mathsf{ct}'_1, \ldots, \mathsf{ct}'_m)) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is over the randomness of KeyGen *and* Enc*.*

Rate-1. *For any* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* KeyGen *and for all ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ *in the support of either* $\mathsf{Enc}(\mathsf{pk}, \cdot)$ *or* $\mathsf{Eval}(\mathsf{pk}, \cdot)$*, it holds that:*

$$\lim_{m \to \infty} \frac{m \cdot |\mathcal{M}|}{|\mathsf{Compress}(\mathsf{pk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_m))|} = 1.$$

That is, the compressed ciphertext output by $\mathsf{Compress}$ *is asymptotically of the same length as the tuple of* m *messages it encrypts.*

8.1 Generic construction from SMS

In Figure 10, we present the construction of rate-1 FHE using a sufficiently powerful SMS scheme. Our construction closely follows the overview from Section 1.1.

Remark 13. We briefly remark that, if the underlying (non-compact) FHE scheme satisfies near-linear decryption (cf. Theorem 4), then we can obtain a rate-1 FHE using the transformation described in Figure 10 and SMS for just degree-2 functions (e.g., succinct NIVOLE, for instance). This is because the decryption can be described as a linear function and we can replace $\mathsf{SMS.Decode}$ with a rounding operation to round-away the error, resulting in additive shares. In other words, we can use the same trick we exploit in our LWE-based construction in Figure 2 and round-away the error from the noisy shares of the decryption using Lemma 1. However, we stress that Figure 10 is *generic* and thus works using any black-box FHE scheme (which may not necessarily have a near-linear decryption).

Rate-1 FHE from SMS

Public Parameters. Let $\mathsf{SMS} = (\mathsf{Setup}, (\mathsf{Encode}_\sigma, \mathsf{Decode}_\sigma)_{\sigma\in\{A,B\}})$ be an SMS scheme, let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an FHE scheme (cf. Definition 2), and f be the function that takes as input any FHE secret key $\widetilde{\mathsf{sk}}$ and any L FHE ciphertexts $\widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_L$, and outputs $\mathsf{FHE.Dec}(\widetilde{\mathsf{sk}}, (\widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_L))$.

$\mathsf{FHE1.KeyGen}(1^\lambda)$:
1 : $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$
3 : $(\mathsf{pe}_B, \mathsf{st}_B) \leftarrow \mathsf{Encode}_B(\mathsf{crs}, f, \mathsf{sk}')$
4 : $\mathsf{pk} := (\mathsf{crs}, \mathsf{pk}', \mathsf{pe}_B)$
5 : $\mathsf{sk} := (\mathsf{crs}, \mathsf{sk}', \mathsf{st}_B)$
6 : **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{FHE1.Compress}(\mathsf{pk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_L))$:
1 : **parse** $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}', \mathsf{pe}_B)$
2 : $X := (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)$
3 : $(\mathsf{pe}_A, \mathsf{st}_A) \leftarrow \mathsf{Encode}_A(\mathsf{crs}, f, X)$
4 : $z_A := \mathsf{Decode}_A(\mathsf{crs}, f, \mathsf{pe}_B, \mathsf{st}_A)$
5 : $\mathsf{ct}^* := (\mathsf{pe}_A, z_A)$
6 : **return** ct^*

$\mathsf{FHE1.CompressDec}(\mathsf{sk}, \mathsf{ct}^*)$:
1 : **parse** $\mathsf{sk} = (\mathsf{crs}, \mathsf{sk}', \mathsf{st}_B)$
2 : **parse** $\mathsf{ct}^* = (\mathsf{pe}_A, z_A)$
3 : $z_B := \mathsf{Decode}_B(\mathsf{crs}, f, \mathsf{pe}_A, \mathsf{st}_B)$
4 : $x := z_A \oplus z_B$
5 : **return** x

$\mathsf{FHE1.Enc}(\mathsf{pk}, x)$:
1 : **parse** $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}', \mathsf{pe}_B)$
2 : **return** $\mathsf{FHE.Enc}(\mathsf{pk}', x)$

$\mathsf{FHE1.Eval}(\mathsf{pk}, f, \mathbf{ct})$:
1 : **parse** $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}', \mathsf{pe}_B)$
2 : **return** $\mathsf{FHE.Eval}(\mathsf{pk}', \mathbf{ct})$

$\mathsf{FHE1.Dec}(\mathsf{sk}, \mathsf{ct})$:
1 : **parse** $\mathsf{sk} = (\mathsf{crs}, \mathsf{sk}', \mathsf{st}_B)$
2 : **return** $\mathsf{FHE.Dec}(\mathsf{sk}', \mathsf{ct})$

Fig. 10. Rate-1 FHE from SMS.

8.2 Security analysis

We now prove security of our construction.

Proposition 5. *Assume that* FHE *is an FHE scheme satisfying Definition 2. The construction of* FHE1 *from Figure 10 is a rate-1 FHE scheme satisfying Definition 19.*

Proof. We prove each required property in turn.

Correctness. Correctness of FHE1.Enc, FHE1.Eval, and FHE1.Dec follow immediately from the correctness of FHE. In fact, our transformation does not modify these algorithms.

Compressed Correctness. We now examine the correctness of FHE1.CompressDec. Notice that from the correctness of SMS, we have that

$$\Pr\Big[\, z_A \oplus z_B = \mathsf{FHE.Dec}(\mathsf{sk}', (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)) \Big] \geq 1 - \mathsf{negl}(\lambda).$$

Separately, by the correctness of FHE and a simple union bound over all $L = L(\lambda) \in \mathsf{poly}(\lambda)$ decryptions, we have that

$$\Pr\Big[\, \mathsf{FHE.Dec}(\mathsf{sk}', (\mathsf{ct}_1, \ldots, \mathsf{ct}_L) = (x_1, \ldots, x_L) \Big] \geq 1 - \mathsf{negl}(\lambda).$$

Therefore, we have that:

$$\Pr\Big[\, z_A \oplus z_B = (x_1, \ldots, x_L) \Big] \geq 1 - \mathsf{negl}(\lambda).$$

This concludes the proof of the compressed correctness property.

Compactness. Similarly to correctness, the compactness follows directly from the compactness property of FHE.

Rate-1. We recall that the compressed ciphertext ct^* output by FHE1.Compress is of the form (pe_A, z_A). Because of the additive reconstruction property of SMS (cf. Definition 4), we have that:

$$|z_A| = |(x_1, \ldots, x_L)| = |\mathcal{M}| \cdot L.$$

Moreover, from the ϵ-succinctness property of SMS (cf. Definition 6), it follows that

$$|\mathsf{pe}_A| \leq |\mathcal{M}| \cdot L^{\epsilon} \cdot \mathsf{poly}(\lambda).$$

Therefore, we have that $|\mathsf{ct}^*| \leq |\mathcal{M}| \cdot L + |\mathcal{M}| \cdot L^{\epsilon} \cdot \mathsf{poly}(\lambda)$, which asymptotically approaches $|\mathcal{M}| \cdot L$.

Security. To prove security, consider the following sequence of hybrids.

- *Hybrid* $\mathcal{H}_0$. This hybrid corresponds to the distribution where x_0 is encrypted. That is, $\mathcal{H}_0$ is defined as:
$$\left\{ (\mathsf{pk}, \mathsf{ct}_0) \;\middle|\; \begin{array}{r} (\mathsf{pk}, \_) \leftarrow \mathsf{FHE1.KeyGen}(1^\lambda) \\ \mathsf{ct}_0 \leftarrow \mathsf{FHE1.Enc}(\mathsf{pk}, x_0) \end{array} \right\}.$$
- *Hybrid* $\mathcal{H}_1$. In this hybrid, we change FHE1.KeyGen to output $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}', \widetilde{\mathsf{pe}}_B)$, where crs and pk are distributed identically to $\mathcal{H}_0$ but where $\widetilde{\mathsf{pe}} \leftarrow \mathsf{Encode}_A(\mathsf{crs}, f, 0)$.
  It follows that $\mathcal{H}_1 \approx_c \mathcal{H}_0$ by the security of SMS.
- *Hybrid* $\mathcal{H}_2$. In this hybrid, we encrypt x_1. That is,
$$\mathcal{H}_2 := \left\{ (\mathsf{pk}, \mathsf{ct}_1) \;\middle|\; \begin{array}{r} (\mathsf{pk}, \_) \leftarrow \mathsf{FHE1.KeyGen}(1^\lambda) \\ \mathsf{ct}_0 \leftarrow \mathsf{FHE1.Enc}(\mathsf{pk}, x_1) \end{array} \right\}.$$
  It follows that $\mathcal{H}_2 \approx_c \mathcal{H}_1$ by the security of FHE, given that FHE1.Enc simply outputs FHE.Enc.
- *Hybrid* $\mathcal{H}_3$. In this hybrid, we revert the changes made in $\mathcal{H}_1$ and output pk distributed identically to KeyGen in $\mathcal{H}_0$.
  It follows that $\mathcal{H}_3 \approx_c \mathcal{H}_2$ by the security of SMS. Note that $\mathcal{H}_3$ is distributed identically to:
$$\left\{ (\mathsf{pk}, \mathsf{ct}_1) \;\middle|\; \begin{array}{r} (\mathsf{pk}, \_) \leftarrow \mathsf{FHE1.KeyGen}(1^\lambda) \\ \mathsf{ct}_1 \leftarrow \mathsf{FHE1.Enc}(\mathsf{pk}, x_1) \end{array} \right\}.$$

It follows that no efficient adversary can distinguish between an encryption of x_0 and x_1 with better than negligible advantage. This concludes the proof of security. ■

9 Correlation-Intractable Hashing from SMS

In this section, we show that an SMS scheme implies a correlation-intractable (CI) hashing for all efficiently searchable relations.

Definition 20 (Searchable Relations [PS19])**.** *We say that a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ is searchable in size S if there exists a function f computable by a size S circuit, such that for any $(x, y) \in R$, it holds that $f(x) = y$. This, in particular, means that for every x, there is at most one witness y for its membership in the relation.*

Definition 21 (Correlation-Intractable Hash Function)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter and let $S(\lambda) \in \mathsf{poly}(\lambda)$ be a circuit size parameter. Let $n = n(\lambda)$ be the input length parameter and $m = m(\lambda)$ be the output length parameter. Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_\lambda$ be a class of relations that is searchable by circuits of size $S(\lambda)$ via functions $\mathcal{F} = \{f_\lambda \colon \{0,1\}^n \to \{0,1\}^m\}_\lambda$. A correlation-intractable (CI) hash function is given by a tuple of algorithms $\mathsf{CI} = (\mathsf{Gen}, \mathsf{Hash})$ with the following syntax:*

- $\mathsf{Gen}(1^\lambda, f) \to \mathsf{hk}$. *The randomized key generation algorithm takes as input the security parameter λ and a function description $f \in \mathcal{F}$. It outputs a hash key hk.*
- $\mathsf{Hash}(\mathsf{hk}, x) \to \mathsf{d}$. *The deterministic hashing algorithm takes as input the hash key and an input $x \in \{0,1\}^n$. It outputs a digest $\mathsf{d} \in \{0,1\}^m$.*

The above algorithms must satisfy the following properties:

Indistinguishability. *For all security parameters $\lambda \in \mathbb{N}$ and all functions $f \in \mathcal{F}$, it holds that:*

$$\left\{ \mathsf{hk} \;\middle|\; \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, f) \right\} \approx_c \left\{ \mathsf{hk} \;\middle|\; \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{0}) \right\},$$

where $\mathbf{0}$ is the all-zeroes function padded to size $S(\lambda)$.

Correlation-Intractability. *A hash function family $(\mathsf{Gen}, \mathsf{Hash})$ is said to be correlation-intractable for the relation $\mathcal{R}$ if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:*

$$\Pr\left[\mathsf{d} = \mathsf{Hash}(\mathsf{hk}, x) \wedge (x, y) \in R_\lambda \quad : \quad \begin{array}{r} \mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, f_\lambda) \\ (x, \mathsf{d}) \leftarrow \mathcal{A}(\mathsf{hk}) \end{array} \right] \leq \mathsf{negl}(\lambda),$$

where the probability is over the randomness of Gen and $\mathcal{A}$.

9.1 Generic construction from SMS

The construction from SMS is nearly identical to the one described by Brakerski et al. [BKM20] using trapdoor hash functions. We note that the digest of the CI hash need not be succinct (and the transformation does not output a succinct digest). Instead, the succinctness property of SMS is used to argue correlation intractability.

Correlation-Intractable Hashing from SMS

Let $\mathsf{SMS} = (\mathsf{Setup}, (\mathsf{Encode}_\sigma, \mathsf{Decode}_\sigma)_{\sigma \in \{A,B\}})$ be an SMS scheme, $\mathcal{F}$ be a family of functions (represented by circuits of size $S = S(\lambda)$) that map $n = n(\lambda)$ bits to $m = m(\lambda)$ bits. Let $\mathcal{U}$ be the a universal circuit that takes as input a function $f \in \mathcal{F}$ and an input x and outputs $f(x)$.

$\mathsf{CI.Gen}(1^\lambda, f)$:

1 : $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{pe}_B, \_) \leftarrow \mathsf{Encode}_B(\mathsf{crs}, \mathcal{U}, f)$
3 : $z \leftarrow\$ \{0,1\}^m$
4 : $r \leftarrow\$ \{0,1\}^\lambda$
5 : $\mathsf{hk} := (\mathsf{crs}, \mathsf{pe}_B, z, r)$
6 : **return** hk

$\mathsf{CI.Hash}(\mathsf{hk}, x)$:

1 : **parse** $\mathsf{hk} = (\mathsf{crs}, \mathsf{st}_B, z, r)$
2 : $(\mathsf{pe}_A, \mathsf{st}_A) \leftarrow \mathsf{Encode}_A(\mathsf{crs}, \mathcal{U}, x; r)$
// Encode_A uses fixed randomness r.
3 : $z_A := \mathsf{Decode}_A(\mathsf{crs}, \mathcal{U}, \mathsf{pe}_B, \mathsf{st}_A)$
4 : $\mathsf{d} := z \oplus z_A$
5 : **return** d

Fig. 11. Correlation-Intractable Hashing from SMS.

Proposition 6. *Assume the existence of an SMS scheme satisfying full succinctness (cf. Definition 6). The* CI *construction from Figure 11 is a correlation-intractable hash function family satisfying Definition 21.*

Proof. We prove each property in turn.

Indistinguishability. The indistinguishability of the hashing keys follows directly from the SMS security of Bob. In particular, the choice of function f is equivalent to Bob's private input in SMS, and indistinguishability follows trivially from Definition 4.

Correlation-Intractability. Suppose, towards contradiction, that there exists an efficient adversary $\mathcal{A}$ that breaks the correlation-intractability property with non-negligible probability $\nu(\lambda)$. That is,

$$\Pr\left[\mathsf{d} = \mathsf{Hash}(\mathsf{hk}, x) \wedge (x, y) \in R_\lambda \quad : \quad \begin{matrix}\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, f_\lambda) \\ (x, \mathsf{d}) \leftarrow \mathcal{A}(\mathsf{hk})\end{matrix}\right] \geq \nu(\lambda).$$

Then, because $(x, y) \in R_\lambda$, we have that $\mathsf{d} = f_\lambda(x)$, and so we can equivalently write $\mathsf{d} = z \oplus z_A$, as defined in Hash. That is, we have that:

$$\Pr\left[f_\lambda(x) = z \oplus z_A \quad : \quad \begin{matrix}\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, f_\lambda) \\ (x, \mathsf{d}) \leftarrow \mathcal{A}(\mathsf{hk})\end{matrix}\right] \geq \nu(\lambda).$$

Next, by the correctness of SMS, we have that $f_\lambda(x) = z_A \oplus \mathsf{Decode}_B(\mathsf{crs}, \mathsf{pe}_A, \mathsf{st}_B)$, and so we have that:

$$\Pr\left[z_A \oplus z_B = z \oplus z_A \quad : \quad \begin{matrix}\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, f_\lambda) \\ (x, \mathsf{d}) \leftarrow \mathcal{A}(\mathsf{hk})\end{matrix}\right] \geq \nu(\lambda) - \mathsf{negl}(\lambda),$$

where $z_B = \mathsf{Decode}_B(\mathsf{crs}, \mathsf{pe}_A, \mathsf{st}_B)$.

Therefore, we have that, with probability at least $\nu(\lambda) - \mathsf{negl}(\lambda)$ over the randomness of Gen and $\mathcal{A}$, it holds that $z = z_B$. We will show that this raises a contradiction.

Let $m = m(\lambda) \in \mathsf{poly}(\lambda)$ be the length of the output of the hash. Note that fixing both crs and st_B, the set of all possible z_B is bounded by the set of all possible pe_B, given that Decode_B is deterministic.

From the ϵ-output succinctness of the SMS scheme, we have that $|\mathsf{pe}_A| \leq m^\epsilon \cdot \mathsf{poly}(\lambda)$. Thus, the set of all possible z_B values has size at most $2^{m^\epsilon \cdot \mathsf{poly}(\lambda)}$. Recall that z is a randomly and independently chosen string of size m. Therefore, the probability that z belongs to this set is at most $\frac{2^{m^\epsilon \cdot \mathsf{poly}(\lambda)}}{2^m}$, which is negligible when m is a large enough polynomial in λ. This contradicts the inequality we derived above, assuming that $\nu(\cdot)$ is a non-negligible function.

This concludes the proof of correlation-intractability and the proof of Proposition 6. ■

Acknowledgments

We thank Geoffroy Couteau for helpful discussions surrounding simulation-based definitions and for the observation captured in Remark 13. We thank Srini Devadas and the anonymous reviewers for helpful comments and suggestions. Elette Boyle was supported, in part, by the AFOSR Award FA9550-21-1-0046 and ERC Project HSS (852952). Abhishek Jain was supported by NSF CNS-1814919, NSF CAREER 1942789, Johns Hopkins University Catalyst award, JP Morgan Faculty Award, and research gifts from Ethereum, Stellar, and Cisco. Akshayaram Srinivasan was supported, in part, by a NSERC Discovery grant RGPIN-2024-03928.

References

ADI+17. B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 223–254. Springer, Cham, August 2017.

ARS24. D. Abram, L. Roy, and P. Scholl. Succinct homomorphic secret sharing. In *EUROCRYPT 2024, Part VI*, *LNCS* 14656, pages 301–330. Springer, Cham, May 2024.

BCG+19. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

BCM+24. D. Bui, G. Couteau, P. Meyer, A. Passelègue, and M. Riahinia. Fast public-key silent OT and more from constrained Naor-Reingold. In *EUROCRYPT 2024, Part VI*, *LNCS* 14656, pages 88–118. Springer, Cham, May 2024.

BDGM19. Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC 2019, Part II*, *LNCS* 11892, pages 407–437. Springer, Cham, December 2019.

BGG+14. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT 2014*, *LNCS* 8441, pages 533–556. Springer, Berlin, Heidelberg, May 2014.

BGI+01. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, *LNCS* 2139, pages 1–18. Springer, Berlin, Heidelberg, August 2001.

BGI14. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *PKC 2014*, *LNCS* 8383, pages 501–519. Springer, Berlin, Heidelberg, March 2014.

BGI15. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In *EUROCRYPT 2015, Part II*, *LNCS* 9057, pages 337–367. Springer, Berlin, Heidelberg, April 2015.

BGI16. E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016, Part I*, *LNCS* 9814, pages 509–539. Springer, Berlin, Heidelberg, August 2016.

BGI+18. E. Boyle, N. Gilboa, Y. Ishai, H. Lin, and S. Tessaro. Foundations of homomorphic secret sharing. In *ITCS 2018*, pages 21:1–21:21. LIPIcs, January 2018.

BGV12. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS 2012*, pages 309–325. ACM, January 2012.

BKM20. Z. Brakerski, V. Koppula, and T. Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In *CRYPTO 2020, Part III*, *LNCS* 12172, pages 738–767. Springer, Cham, August 2020.

BKS19. E. Boyle, L. Kohl, and P. Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT 2019, Part II*, *LNCS* 11477, pages 3–33. Springer, Cham, May 2019.

BL18. F. Benhamouda and H. Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *EUROCRYPT 2018, Part II*, *LNCS* 10821, pages 500–532. Springer, Cham, April / May 2018.

BLW17. D. Boneh, K. Lewi, and D. J. Wu. Constraining pseudorandom functions privately. In *PKC 2017, Part II*, *LNCS* 10175, pages 494–524. Springer, Berlin, Heidelberg, March 2017.

BOV03. B. Barak, S. J. Ong, and S. P. Vadhan. Derandomization in cryptography. In *CRYPTO 2003*, *LNCS* 2729, pages 299–315. Springer, Berlin, Heidelberg, August 2003.

BV11. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

BW13. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT 2013, Part II*, *LNCS* 8270, pages 280–300. Springer, Berlin, Heidelberg, December 2013.

BZ14. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO 2014, Part I*, *LNCS* 8616, pages 480–499. Springer, Berlin, Heidelberg, August 2014.

CCH+19. R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, R. D. Rothblum, and D. Wichs. Fiat-Shamir: from practice to theory. In *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

CDD+24. G. Couteau, L. Devadas, S. Devadas, A. Koch, and S. Servan-Schreiber. QuietOT: Lightweight oblivious transfer with a public-key setup. In *ASIACRYPT 2024, Part II, LNCS 15485*, pages 197–231. Springer, Singapore, 2024.

CDG+17. C. Cho, N. Döttling, S. Garg, D. Gupta, P. Miao, and A. Polychroniadou. Laconic oblivious transfer and its applications. In *CRYPTO 2017, Part II*, *LNCS* 10402, pages 33–65. Springer, Cham, August 2017.

CGH98. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

CGV15. A. Cohen, S. Goldwasser, and V. Vaikuntanathan. Aggregate pseudorandom functions and connections to learning. In *TCC 2015, Part II*, *LNCS* 9015, pages 61–89. Springer, Berlin, Heidelberg, March 2015.

CHK+19. A. R. Choudhuri, P. Hubácek, C. Kamath, K. Pietrzak, A. Rosen, and G. N. Rothblum. Finding a nash equilibrium is no easier than breaking Fiat-Shamir. In *51st ACM STOC*, pages 1103–1114. ACM Press, June 2019.

CJJ21. A. R. Choudhuri, A. Jain, and Z. Jin. Non-interactive batch arguments for NP from standard assumptions. In *CRYPTO 2021, Part IV*, *LNCS* 12828, pages 394–423, Virtual Event, August 2021. Springer, Cham.

CJJ22. A. R. Choudhuri, A. Jain, and Z. Jin. SNARGs for $\mathcal{P}$ from LWE. In *62nd FOCS*, pages 68–79. IEEE Computer Society Press, February 2022.

CKS08. D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. In *EUROCRYPT 2008*, *LNCS* 4965, pages 127–145. Springer, Berlin, Heidelberg, April 2008.

CMPR23. G. Couteau, P. Meyer, A. Passelègue, and M. Riahinia. Constrained pseudorandom functions from homomorphic secret sharing. In *EUROCRYPT 2023, Part III*, *LNCS* 14006, pages 194–224. Springer, Cham, April 2023.

DGI+19. N. Döttling, S. Garg, Y. Ishai, G. Malavolta, T. Mour, and R. Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 3–32. Springer, Cham, August 2019.

DH76. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DHRW16. Y. Dodis, S. Halevi, R. D. Rothblum, and D. Wichs. Spooky encryption and its applications. In *CRYPTO 2016, Part III*, *LNCS* 9816, pages 93–122. Springer, Berlin, Heidelberg, August 2016.

FHKP13. E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In *PKC 2013*, *LNCS* 7778, pages 254–271. Springer, Berlin, Heidelberg, February / March 2013.

FKN94. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation (extended abstract). In *26th ACM STOC*, pages 554–563. ACM Press, May 1994.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, *LNCS* 263, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

Gen09. C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GGH+13. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GH19. C. Gentry and S. Halevi. Compressible FHE with applications to PIR. In *TCC 2019, Part II*, *LNCS* 11892, pages 438–464. Springer, Cham, December 2019.

GHO20. S. Garg, M. Hajiabadi, and R. Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 OT and more. In *TCC 2020, Part I*, *LNCS* 12550, pages 88–116. Springer, Cham, November 2020.

GKM+00. Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335. IEEE Computer Society Press, November 2000.

GKPV10. S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS 2010*, pages 230–240. Tsinghua University Press, January 2010.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GS18. S. Garg and A. Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT 2018, Part II*, *LNCS* 10821, pages 468–499. Springer, Cham, April / May 2018.

GSW13. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 75–92. Springer, Berlin, Heidelberg, August 2013.

GVW15. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 503–523. Springer, Berlin, Heidelberg, August 2015.

HLL23. Y.-C. Hsieh, H. Lin, and J. Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023.

HLP11. S. Halevi, Y. Lindell, and B. Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *CRYPTO 2011*, *LNCS* 6841, pages 132–150. Springer, Berlin, Heidelberg, August 2011.

HW15. P. Hubacek and D. Wichs. On the communication complexity of secure function evaluation with long output. In *ITCS 2015*, pages 163–172. ACM, January 2015.

JJ21. A. Jain and Z. Jin. Non-interactive zero knowledge from sub-exponential DDH. In *EUROCRYPT 2021, Part I*, *LNCS* 12696, pages 3–32. Springer, Cham, October 2021.

JKKZ21. R. Jawale, Y. T. Kalai, D. Khurana, and R. Y. Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In *53rd ACM STOC*, pages 708–721. ACM Press, June 2021.

JLS21. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. In *53rd ACM STOC*, pages 60–73. ACM Press, June 2021.

KLVW23. Y. Kalai, A. Lombardi, V. Vaikuntanathan, and D. Wichs. Boosting batch arguments and RAM delegation. In *55th ACM STOC*, pages 1545–1552. ACM Press, June 2023.

KPTZ13. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, *LNCS* 7237, pages 700–718. Springer, Berlin, Heidelberg, April 2012.

OPWW15. T. Okamoto, K. Pietrzak, B. Waters, and D. Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In *ASIACRYPT 2015, Part I*, *LNCS* 9452, pages 121–145. Springer, Berlin, Heidelberg, November / December 2015.

OSY21. C. Orlandi, P. Scholl, and S. Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT 2021, Part I*, *LNCS* 12696, pages 678–708. Springer, Cham, October 2021.

PS19. C. Peikert and S. Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO 2019, Part I*, *LNCS* 11692, pages 89–114. Springer, Cham, August 2019.

QWW18. W. Quach, H. Wee, and D. Wichs. Laconic function evaluation and applications. In *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

Reg05. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

SW14. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

Yao86. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

Supplementary Material

A Generic Upgrade to a Simulation-Based Definition

In this section, we show that Definition 4 can be generically upgraded to satisfy a simulation-based definition for secure computation. Specifically, we show that any SMS scheme can be generically made to satisfy the (corruptible) ideal functionality presented in Functionality 1.

Functionality $\mathcal{F}_{\mathsf{sms}}$

Procedure. The functionality is instantiated between parties Alice and Bob and an adversary $\mathcal{A}$ playing the role of Charlie and possibly corrupting either Alice or Bob. The functionality aborts if it receives any incorrectly formatted messages.

- If both parties are honest:
  1: Wait for input (X, f) from Alice and (y, f) from Bob.
  2: Sample $R_i \leftarrow\$ \{0,1\}^{|f(\cdot,\cdot)|}$.
  3: Output $f(X, y) \oplus R_i$ to Alice, R_i to Bob, and $(f(X, y) \oplus R_i, R_i)$ to $\mathcal{A}$.
- If Alice is corrupted:
  1: Wait for input (X, f, out) from $\mathcal{A}$ and (y, f) from Bob.
  2: Output $(f(X, y) \oplus \mathsf{out},\ \mathsf{out})$ to $\mathcal{A}$, $f(X, y) \oplus \mathsf{out}$ to Bob.
- If Bob is corrupted:
  1: Wait for input (y, out) from $\mathcal{A}$ and X from Alice.
  2: Output $(f(X, y) \oplus \mathsf{out},\ \mathsf{out})$ to $\mathcal{A}$, $f(X, y) \oplus \mathsf{out}$ to Alice.

Functionality 1. Corruptible ideal functionality for SMS.

Transformation. To transform any SMS scheme satisfying Definition 4 into one that instantiates the ideal functionality presented in Functionality 1, we need to "re-randomize" the output shares (otherwise the simulator would be unable to properly simulate the output in the case where both parties are honest). This randomization can be achieved with the help of a pseudorandom function that the parties use to generate pseudorandom shares of zero, and does not require introducing any new assumptions. The transformation is described in Figure 12 and uses a non-interactive key exchange (NIKE), which we recall is implied by Definition 4 using the reduction of Boyle et al. [BGI+18]. We formally define NIKE in Definition 22.

Definition 22 (Non-Interactive Key Exchange [DH76, CKS08, FHKP13])**.** *Let $\lambda \in \mathbb{N}$ be a security parameter. A non-interactive key exchange (NIKE) scheme consists of algorithms* $\mathsf{NIKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDer})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$. *The randomized setup algorithm takes as input the security parameter λ and outputs a common reference string* crs.
- $\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$. *The randomized key generation algorithm takes as input the CRS* crs. *It outputs a public key* pk *and secret key* sk.
- $\mathsf{KeyDer}(\mathsf{crs}, \mathsf{pk}_i, \mathsf{sk}_j) \to K$. *The deterministic key derivation algorithm takes as input the CRS* crs, *a public key* pk_i, *and a secret key* sk_j. *It outputs a key $K \in \{0,1\}^\lambda$.*

The above algorithms must satisfy the following properties:

Correctness. *For all security parameters $\lambda \in \mathbb{N}$, it holds that:*

$$\Pr\left[\begin{array}{rcl} & & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ & & (\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{KeyGen}(\mathsf{crs}) \\ K_A = K_B & : & (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{KeyGen}(\mathsf{crs}) \\ & & K_A \leftarrow \mathsf{KeyDer}(\mathsf{crs}, \mathsf{pk}_B, \mathsf{sk}_A) \\ & & K_B \leftarrow \mathsf{KeyDer}(\mathsf{crs}, \mathsf{pk}_A, \mathsf{sk}_B) \end{array}\right] = 1.$$

Security. *For all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:*

$$\Pr\left[\begin{array}{rcl} & & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ & & (\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{KeyGen}(\mathsf{crs}) \\ & & (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{KeyGen}(\mathsf{crs}) \\ b = b' & : & K_0 \leftarrow \mathsf{KeyDer}(\mathsf{crs}, \mathsf{pk}_A, \mathsf{sk}_B) \\ & & K_1 \leftarrow\!\!\$\, \{0,1\}^\lambda \\ & & b \leftarrow\!\!\$\, \{0,1\} \\ & & b' \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{pk}_A, \mathsf{pk}_B, K_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

In particular, this security definition for NIKE is known as "CKS-light" security [FHKP13], which is known to be polynomially equivalent to stronger notions of NIKE.

Simulation-Secure SMS Transformation

Let $\mathsf{SMS} = (\mathsf{Setup}, (\mathsf{Encode}_\sigma, \mathsf{Decode}_\sigma)_{\sigma\in\{A,B\}})$ be an SMS scheme, $\mathsf{NIKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDer})$ be a NIKE scheme, and let $F\colon \{0,1\}^\lambda \times \{0,1\}^\star \to \{0,1\}^m$ be a PRF. We let $\bar{\sigma} := \{A, B\} \setminus \{\sigma\}$.

$\mathsf{SMS}^*.\mathsf{Setup}(1^\lambda)$:

1 : $\mathsf{crs}_{\mathsf{sms}} \leftarrow \mathsf{SMS.Setup}(1^\lambda)$
2 : $\mathsf{crs}_{\mathsf{nike}} \leftarrow \mathsf{NIKE.Setup}(1^\lambda)$
3 : **return** $\mathsf{crs} := (\mathsf{crs}_{\mathsf{sms}}, \mathsf{crs}_{\mathsf{nike}})$

$\mathsf{SMS}^*.\mathsf{Encode}_\sigma(\mathsf{crs}, f, x_\sigma)$:

1 : **parse** $\mathsf{crs} = (\mathsf{crs}_{\mathsf{sms}}, \mathsf{crs}_{\mathsf{nike}})$
2 : $(\mathsf{pe}'_\sigma, \mathsf{st}'_\sigma) \leftarrow \mathsf{SMS.Encode}_\sigma(\mathsf{crs}, f, x_\sigma)$
3 : $(\mathsf{pk}_\sigma, \mathsf{sk}_\sigma) \leftarrow \mathsf{NIKE.KeyGen}(\mathsf{crs}_{\mathsf{nike}})$
4 : $\mathsf{pe}_\sigma := (\mathsf{pe}'_\sigma, \mathsf{pk}_\sigma)$
5 : **return** $(\mathsf{pe}_\sigma, \mathsf{st}_\sigma)$

$\mathsf{SMS}^*.\mathsf{Decode}_\sigma(\mathsf{crs}, f, \mathsf{pe}_{\bar{\sigma}}, \mathsf{st}_\sigma)$:

1 : **parse** $\mathsf{crs} = (\mathsf{crs}_{\mathsf{sms}}, \mathsf{crs}_{\mathsf{nike}})$
2 : **parse** $\mathsf{pe}_{\bar{\sigma}} = (\mathsf{pe}'_{\bar{\sigma}}, \mathsf{pk}_{\bar{\sigma}})$
3 : $z'_\sigma := \mathsf{SMS.Decode}_\sigma(\mathsf{crs}, \mathsf{pe}'_{\bar{\sigma}}, \mathsf{st}_\sigma)$
4 : $K := \mathsf{NIKE.KeyDer}(\mathsf{pk}_{\bar{\sigma}}, \mathsf{sk}_\sigma)$
5 : $z_\sigma := z'_\sigma \oplus F_K(f)$

Fig. 12. Generic transformation to simulation-security.

Claim. The transformed SMS scheme SMS^* described in Figure 12, when viewed as an interactive protocol between Alice, Bob, and Charlie, securely instantiates the ideal functionality $\mathcal{F}_{\mathsf{sms}}$.

Proof. We consider the three possible cases:

Case 1: Both parties are honest. On input the CRS crs and $f(X, y)$, the simulator generates z_A uniformly at random and defines $z_B := z_A \oplus f(X, y)$ and outputs (z_A, z_B) as the simulated view of the adversary, which matches the output of $\mathcal{F}_{\mathsf{sms}}$. We prove that this simulated view is computationally indistinguishable to the real view of the adversary via a hybrid argument:

– *Hybrid* $\mathcal{H}_0$. This hybrid corresponds to the output (z_A, z_B) in the real view.

– *Hybrid* $\mathcal{H}_1$. In this hybrid, we define $z_B := z_A \oplus f(X, y)$. This hybrid is statistically indistinguishable to the previous one by the correctness property of the underlying SMS scheme.

– *Hybrid* $\mathcal{H}_2$. In this hybrid, the key K is sampled uniformly at random. This hybrid is computationally indistinguishable to the previous one by the security of the NIKE scheme.

– *Hybrid* $\mathcal{H}_3$. In this hybrid, z_A is sampled uniformly at random from $\{0,1\}^m$. This hybrid is computationally indistinguishable to the previous one by the pseudorandomness of the PRF.

At this point, it suffices to note that the distribution in $\mathcal{H}_3$ is identical to the simulated distribution, concluding the proof.

Case 2: Party A is corrupted. Suppose that Alice is corrupted by the adversary $\mathcal{A}$. We construct an efficient simulator $\mathcal{S}$ that interacts with $\mathcal{F}_{\mathsf{sms}}$ to simulate the view of the adversary $\mathcal{A}$. We start with the description of $\mathcal{S}$.

$\mathcal{S}$: On input the CRS crs, Alice's input (X, f), and the random coins of $\mathcal{A}$,
- Compute $(\mathsf{pe}_B, \_) \leftarrow \mathsf{SMS.Encode}_B(\mathsf{crs}, 0)$.
- Use the private state st_A of Alice (which is can be obtained from the random coins of $\mathcal{A}$ in the semi-honest setting) to compute $z_A := \mathsf{Decode}_A(\mathsf{crs}, f, \mathsf{pe}_B, \mathsf{st}_A)$.
- Send (X, f, z_A) to $\mathcal{F}_{\mathsf{sms}}$ on behalf of $\mathcal{A}$, and receive $(f(X, y) \oplus z_A, z_A)$ from $\mathcal{F}_{\mathsf{sms}}$.
- Define $z_B := f(X, y) \oplus z_A$.
- Output $(X, f, z_A, \mathsf{pe}_B, z_B)$.

We now show that the view generated by $\mathcal{S}$ is computationally indistinguishable to the view of $\mathcal{A}$ in the real protocol execution via the following sequence of hybrids.

– *Hybrid* $\mathcal{H}_0$. This hybrid consists of the view $(X, f, z_A, \mathsf{pe}_B, z_B)$ of the adversary $\mathcal{A}$ in the real execution of the protocol, where $\mathcal{A}$ corrupts Alice and plays the role of Charlie.

– *Hybrid* $\mathcal{H}_1$. In this hybrid, we define z_B as $z_B := z_A \oplus f(X, y)$. This hybrid is statistically close to the previous one by the correctness property of the SMS scheme.

– *Hybrid* $\mathcal{H}_2$. In this hybrid, we generate pe_B as the output of $\mathsf{SMS.Encode}_B(\mathsf{crs}, 0)$ and derive z_A using $\mathsf{SMS.Decode}_A$. This hybrid is indistinguishable to the previous one from the security property of the SMS scheme.

At this point, it suffices to note that $\mathcal{H}_2$ is distributed identically to the view generated by $\mathcal{S}$ in the ideal world, which concludes the proof.

Case 3: Party B is corrupted. This case follows by symmetry.

■

B Additional Preliminaries

B.1 Constrained PRFs

Definition 23 (Constrained Pseudorandom Functions; Adapted from [BW13, CMPR23])**.** *Let $\lambda \in \mathbb{N}$ be a security parameter. A Constrained Pseudorandom Function (CPRF) with key space $\mathcal{K} = \mathcal{K}_\lambda$, domain $\mathcal{X} = \mathcal{X}_\lambda$, and range $\mathcal{Y}$, that supports constraints represented by the class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where $C_\lambda \colon \mathcal{X} \to \{0,1\}$, consists of the following four algorithms.*

– $\mathsf{KeyGen}(1^\lambda) \to \mathsf{msk}$. *The randomized key generation algorithm takes as input a security parameter λ. Outputs a master secret key $\mathsf{msk} \in \mathcal{K}$.*

– $\mathsf{Eval}(\mathsf{msk}, x) \to y$. *The deterministic evaluation algorithm takes as input the master secret key msk and input $x \in \mathcal{X}$. Outputs $y \in \mathcal{Y}$.*

– $\mathsf{Constrain}(\mathsf{msk}, C) \to \mathsf{csk}$. *The randomized constrain algorithm takes as input the master secret key msk and a constraint circuit $C \in \mathcal{C}$. Outputs a constrained key csk.*

– $\mathsf{CEval}(\mathsf{csk}, x) \to y$. *The deterministic contained evaluation algorithm takes as input the constrained key* csk *and an input* $x \in \mathcal{X}$. *Outputs* $y \in \mathcal{Y}$.

We let any auxiliary public parameters PP *be an implicit input to all algorithms. A CPRF must satisfy the following correctness and security properties.*

Correctness. *For all security parameters* λ, *all constraints* $C \in \mathcal{C}$, *and all inputs* $x \in \mathcal{X}$ *such that* $C(x) = 0$ *(authorized), it holds that:*

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{CEval}(\mathsf{csk}, x) \quad : \quad \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

(1-key, adaptive) Security. *A CPRF is (1-key, adaptively)-secure if for all efficient adversaries* $\mathcal{A}$, *the advantage of* $\mathcal{A}$ *in the following security experiment* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},b}(\lambda)$ *is negligible in* λ. *Here,* b *denotes the challenge bit.*

1. ***Setup:*** *On input* 1^λ, *the challenger runs* $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, *initializes the set* $Q := \emptyset$, *and runs* $\mathcal{A}(1^\lambda)$.
2. ***Pre-challenge queries:*** $\mathcal{A}$ *adaptively sends arbitrary inputs* $x \in \mathcal{X}$ *to the challenger. For each* x, *the challenger computes* $y := \mathsf{Eval}(\mathsf{msk}, x)$, *sends* y *to* $\mathcal{A}$, *and proceeds to update* $Q := Q \cup \{x\}$.
3. ***Constrain query:*** $\mathcal{A}$ *sends* one *constraint* $C \in \mathcal{C}$ *to the challenger. The challenger computes* $\mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$, *and sends* csk *to* $\mathcal{A}$.
4. ***Challenge query:*** *For the single challenge query,* $\mathcal{A}$ *sends input* $x^* \in \mathcal{X}$ *as its challenge query, subject to the restriction that* $x^* \notin Q$ *and* $C(x^*) \neq 0$. *If* $b = 0$, *the challenger computes* $y^* := \mathsf{Eval}(\mathsf{msk}, x^*)$. *Else, if* $b = 1$, *the challenger picks* $y^* \leftarrow\!\!\$\, \mathcal{Y}$. *The challenger sends* y^* *to* $\mathcal{A}$.
5. ***Post-challenge queries:*** $\mathcal{A}$ *continues to adaptively query the challenger on inputs* $x \in \mathcal{X}$, *subject to the restriction that* $x \neq x^*$. *For each* x, *the challenger computes* $y := \mathsf{Eval}(\mathsf{msk}, x)$ *and sends* y *to* $\mathcal{A}$.
6. ***Guess:*** $\mathcal{A}$ *outputs its guess* b', *which is the output of the experiment.*

$\mathcal{A}$ *wins if* $b' = b$, *and its advantage* $\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda)$ *is defined as*

$$\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},1}(\lambda) = 1]\right|,$$

where the probability is over the randomness of $\mathcal{A}$ *and* KeyGen *and* $\mathsf{Constrain}$.

C Deferred Proofs

C.1 Proof of Proposition 3

We prove that the real view of the adversary is computationally indistinguishable to a simulated view. First, we describe the simulator $\mathcal{S}$ for pe_B.

$\mathcal{S}$: On input crs,

– Parse $\mathsf{crs} = \mathsf{hk}$.
– Sample $K \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$
– Compute $\widetilde{P^{\mathsf{sim}}} \leftarrow \mathsf{iO}(1^\lambda, P^{\mathsf{sim}})$, where P^{sim} is as described in Program 4 with hardcoded inputs (hk, K). Notice that the program does not contain y.
– Output $\mathsf{pe}_B := \widetilde{P^{\mathsf{sim}}}$

Program 4: The Simulated Program

Hardcoded: (hk, K).
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**
2: $\quad R_i := \mathsf{PPRF.Eval}(K, c_{\widehat{X}} \| f \| i)$
3: $\quad$ **return** R_i
4: **else return** $\perp$

We now turn to proving that the output of $\mathcal{S}$ is computationally indistinguishable to pe_B as computed by Encode_B in Figure 5.

Notation. Let n denote the domain length (in bits) of the puncturable PRF PPRF and consider the 2^n possible inputs to PPRF. Let $(c_j \| f_j \| i_j)$, parsed as a binary string of length n, denote the j-th canonical input in the domain of the PPRF.

Circuit padding. We assume, without loss of generality, that all obfuscated programs (including Bob's Program 1 and the simulated Program 4) have a polynomial amount of padding added to the circuit so as to make all the obfuscations used in the security proof have the same circuit size as Program 1.

Consider the following sequence of hybrids.

- *Hybrid* $\mathcal{H}_0$. This hybrid consists of the pe_B computed exactly according to Encode_B in Figure 5. In particular, pe_B consists of an obfuscation of program P described in Program 1.

- *Hybrid* $\mathcal{H}_{1,j}$. We define $\mathcal{H}_{1,j}$ to be the hybrid distribution where we replace the obfuscation of the program P with an obfuscation of the program $P^{(1,j)}_{\mathsf{hyb}}$. The program $P^{(1,j)}_{\mathsf{hyb}}$ is described in Hybrid 1. Moreover, in this hybrid, we set $\mathsf{hk} \leftarrow \mathsf{SSB.Gen}(1^\lambda, L, i_j)$. That is, we make the SSB hash statistically binding on index i_j as parsed from the j-th canonical input $(c_j \| f_j \| i_j)$.
  $P^{(1,j)}_{\mathsf{hyb}}$ has a PPRF master key K_0, along with the j-th canonical input in the PPRF domain (denoted $(c_j \| f_j \| i_j)$), as additional harcoded inputs. Additionally, it uses the output mask R_i, computed as $R_i := \mathsf{PPRF.Eval}(K_0, c_{\widehat{X}} \| f \| i)$, for all inputs $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$ where $(c_{\widehat{X}} \| f \| i)$ is smaller than $(c_j \| f_j \| i_j)$ (the comparison is performed with respect to some arbitrary total ordering assigned to all the inputs in the PPRF domain).

Hybrid 1: (Parameterized by $j \in \{0, 1, \ldots, 2^n\}$ and a universal circuit $\mathcal{U}$)

Hardcoded: $(\mathsf{hk}, K_0, K, y, (c_j \| f_j \| i_j))$.
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

if $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**

if $(c_{\widehat{X}} \| f \| i) < (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K_0, c_{\widehat{X}}, f, i)$
2: **return** R

if $(c_{\widehat{X}} \| f \| i) \geq (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K, c_{\widehat{X}} \| f \| i)$
2: $d := \mathcal{U}(f, x_i, y)$
3: **return** $d \oplus R_i$

else return $\perp$

Claim. $\mathcal{H}_{1,0} \approx_c \mathcal{H}_0$ assuming the security of $i\mathcal{O}$.

Proof. The only difference between $\mathcal{H}_{1,0}$ and $\mathcal{H}_0$ is the inclusion of additional hardcoded inputs since the PPRF key K_0 is not used when $j = 0$. In particular, $P^{(1,0)}_{\mathsf{hyb}}$ and P are functionally equivalent (and of equivalent size due to padding). Indistinguishability thus follows directly from the security of $i\mathcal{O}$. □

- *Hybrid* $\mathcal{H}_{2,j}$. We define $\mathcal{H}_{2,j}$ to be the hybrid distribution where we replace the obfuscation of the program $P^{(1,j)}_{\mathsf{hyb}}$ with an obfuscation of the program $P^{(2,j)}_{\mathsf{hyb}}$. The program $P^{(2,j)}_{\mathsf{hyb}}$ is described in Hybrid 2 and has the hardcoded master PPRF key K replaced with a punctured PPRF key $K^* \leftarrow \mathsf{PPRF.Puncture}(K, (c_j \| f_j \| i_j))$, and additionally has the value $R^* := \mathsf{PPRF.Eval}(K, c_j \| f_j \| i_j)$ as a hardcoded input.

Hybrid 2: (Parameterized by $j \in \{0, 1, \ldots, 2^n\}$ and a universal circuit $\mathcal{U}$)

Hardcoded: $(\mathsf{hk}, K_0, K^*, R^*, y, (c_j \| f_j \| i_j))$.
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**

if $(c_{\widehat{X}} \| f \| i) < (c_j \| f_j \| i_j)$	**if** $(c_{\widehat{X}} \| f \| i) = (c_j \| f_j \| i_j)$	**if** $(c_{\widehat{X}} \| f \| i) > (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K_0, c_{\widehat{X}} \| f \| i)$	1: $d := \mathcal{U}(f, x_i, y)$	1: $R_i := \mathsf{PPRF.Eval}(K^*, c_{\widehat{X}} \| f \| i)$
2: **return** R_i	2: **return** $d \oplus R^*$	2: $d := \mathcal{U}(f, x_i, y)$
		3: **return** $d \oplus R_i$

2: **else return** $\perp$

Claim. $\mathcal{H}_{2,j} \approx_c \mathcal{H}_{1,j}$ assuming the security of $i\mathcal{O}$.

Proof. Note that the program $P^{(2,j)}_{\mathsf{hyb}}$ outputs the same mask value as the program $P^{(1,j)}_{\mathsf{hyb}}$ on the punctured PPRF input $(c_j \| f_j \| i_j)$, given that $R^* = \mathsf{PPRF.Eval}(K, c_j \| f_j \| i_j)$. Furthermore, since $\mathsf{PPRF.Eval}(K, \cdot)$ and $\mathsf{PPRF.Eval}(K^*, \cdot)$ agree on all other inputs, the two programs are functionally equivalent. The claim then follows directly from the security of $i\mathcal{O}$. □

– *Hybrid* $\mathcal{H}_{3,j}$. We define $\mathcal{H}_{3,j}$ to be the hybrid distribution where we replace the obfuscation of the program $P^{(2,j)}_{\mathsf{hyb}}$ with an obfuscation of the program $P^{(3,j)}_{\mathsf{hyb}}$. The program $P^{(3,j)}_{\mathsf{hyb}}$ is described in Hybrid 3 and has the hardcoded mask R^* replaced with a uniformly random output R sampled from the support of the PPRF.

Hybrid 3: (Parameterized by $j \in \{0, 1, \ldots, 2^n\}$ and a universal circuit $\mathcal{U}$)

Hardcoded: $(\mathsf{hk}, K_0, K^*, R, y, (c_j \| f_j \| i_j))$.
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**

if $(c_{\widehat{X}} \| f \| i) < (c_j \| f_j \| i_j)$	**if** $(c_{\widehat{X}} \| f \| i) = (c_j \| f_j \| i_j)$	**if** $(c_{\widehat{X}} \| f \| i) > (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K_0, c_{\widehat{X}} \| f \| i)$	1: $d := \mathcal{U}(f, x_i, y)$	1: $R_i := \mathsf{PPRF.Eval}(K^*, c_{\widehat{X}} \| f \| i)$
2: **return** R_i	2: **return** $d \oplus R$	2: $d := \mathcal{U}(f, x_i, y)$
		3: **return** $d \oplus R_i$

2: **else return** $\perp$

Claim. $\mathcal{H}_{3,j} \approx_c \mathcal{H}_{2,j}$ assuming the security of the PPRF.

Proof. Notice that any distinguisher between $\mathcal{H}_{3,j} \approx_c \mathcal{H}_{2,j}$ is also a distinguisher for the PPRF security game, given that R^* is distributed identically to the case where the challenger outputs the PPRF evaluation and R is distributed identically to the case where the challenger outputs a uniformly random output. The claim then follows from the security of the PPRF. □

– *Hybrid* $\mathcal{H}_{4,j}$. This hybrid depends on a preprocessing phase.

1. In the preprocessing phase, the value x_{i_j} is computed by finding any tuple of values $(\widehat{x}_{i_j}, x_{i_j}, \pi_i, r_i)$ such that:
$$\widehat{x}_{c_j} = \mathsf{Commit}(x_{i_j}; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_j, \widehat{x}_{i_j}, \pi_{i_j}) = 1.$$
Then, the value d_{i_j} is computed by evaluating $d_{i_j} := \mathcal{U}(f_j, x_{i_j}, y)$.

2. We then define $\mathcal{H}_{4,j}$ to be the hybrid distribution where we replace the obfuscation of the program $P^{(3,j)}_{\mathsf{hyb}}$ with an obfuscation of the program $P^{(4,j)}_{\mathsf{hyb}}$. The program $P^{(4,j)}_{\mathsf{hyb}}$ is described in Hybrid 4 and has the value $d_j \oplus R$ hardcoded as an input, where R is as defined in $\mathcal{H}_{3,j}$.

Hybrid 4: (Parameterized by $j \in \{0, 1, \ldots, 2^n\}$ and a universal circuit $\mathcal{U}$)

Hardcoded: $(\mathsf{hk}, K_0, K^*, d_j \oplus R, y, (c_j \| f_j \| i_j))$.
Input: $(c_{\widehat{x}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{x}}, \widehat{x}_i, i, \pi_i) = 1$ **then**

if $(c_{\widehat{x}}, f, i) < (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K_0, c_{\widehat{x}} \| f \| i)$
2: **return** R_i

if $(c_{\widehat{x}}, f, i) = (c_j \| f_j \| i_j)$
1: **return** $d_j \oplus R$

if $(c_{\widehat{x}} \| f \| i) > (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K^*, c_{\widehat{x}} \| f \| i)$
2: $d := \mathcal{U}(f, x_i, y)$
3: **return** $d \oplus R_i$

2: **else return** $\perp$

Claim. $\mathcal{H}_{4,j} \approx_c \mathcal{H}_{3,j}$ assuming the security of $i\mathcal{O}$, the somewhere *perfect* binding of the SSB hash function, and the perfect binding of the commitment scheme.

Proof. At a high level, we prove that there is only one input to the program $P^{(3,j)}_{\mathsf{hyb}}$ that makes it output $d \oplus R$ at the punctured input. Then, because $d_j \oplus R$ is hardcoded in $P^{(3,j)}_{\mathsf{hyb}}$, and is identical to the output on the punctured input in $P^{(4,j)}_{\mathsf{hyb}}$, we can invoke the security of $i\mathcal{O}$ to finish proving the claim.

Formally, suppose, towards contradiction, that there exists a pair of inputs on which $P^{(3,j)}_{\mathsf{hyb}}$ outputs $d \oplus R$ and $d' \oplus R$, respectively, using the same hardcoded value of R and some $d \neq d'$.[13] Let this pair of inputs be:

$$(c_{\widehat{x}}, x, (\widehat{x}, r), i, \pi, f) \neq (c'_{\widehat{x}}, x', (\widehat{x}', r'), i', \pi', f').$$

By inspection of $P^{(3,j)}_{\mathsf{hyb}}$ (described in Hybrid 3), it is clear that if these inputs produce outputs $d \oplus R$ and $d' \oplus R$, respectively, then the following three conditions must hold simultaneously:

(1) $(c_{\widehat{x}}, f, i) = (c_j, f_j, i_j) = (c'_{\widehat{x}}, f', i')$, // Otherwise, R is not used.
(2) $\widehat{x} = \mathsf{Commit}(x; r) \wedge \widehat{x}' = \mathsf{Commit}(x'; r')$, and // Otherwise, the output is $\perp$.
(3) $\mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{x}}, \widehat{x}, i, \pi) = \mathsf{SSB.Verify}(\mathsf{hk}, c'_{\widehat{x}}, \widehat{x}', i', \pi') = 1$.

By (1) we have that $f = f'$, and so it must be the case that $x \neq x'$ given that $d = f(x, y) \neq f(x', y) = d'$. Moreover, because Commit is perfectly-binding, we also have that $\widehat{x} \neq \widehat{x}'$. Then, using (1), we can rewrite (3) as:

$$\mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{x}}, \widehat{x}, i, \pi) = \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{x}}, \widehat{x}', i, \pi') = 1.$$

This implies that there exist at least two openings $\widehat{x}' \neq \widehat{x}$ for the same index i, such that $\mathsf{SSB.Verify}$ accepts under the same hash key hk and hash $c_{\widehat{x}}$. However, in $P^{(3,j)}_{\mathsf{hyb}}$, the SSB hash hk was set to be perfectly binding on index $i_j = i$, which raises a contradiction. Intuitively, the PPRF evaluation forces $c_{\widehat{x}}$ and i to be consistent with $c'_{\widehat{x}}$ and i'.

At this point, we conclude that the hardcoded value R is only used to mask a single output value d in $P^{(3,j)}_{\mathsf{hyb}}$. Moreover, by the analysis above, the output value d_j (as computed in the preprocessing phase of $P^{(4,j)}_{\mathsf{hyb}}$) must be equal to d (as output by $P^{(3,j)}_{\mathsf{hyb}}$). To see this, first note that $c_{\widehat{x}} = c_j$ since otherwise d is not output. Then, by our analysis above, we have that the inputs $x_{i_j} = x$

[13] While there may be other inputs that produce collisions in the PPRF outputs such that $R_i = R$, we only need to examine the case where the *hardcoded* value R is used twice. For all other inputs, the two programs behave identically.

and $f_j = f$ are uniquely determined by $c_j = c_{\widehat{X}}$. So the preprocessing outputs $d_j = d$, as output by $P^{(3,j)}_{\mathsf{hyb}}$ on the punctured input.

It follows that $P^{(4,j)}_{\mathsf{hyb}}$ is functionally equivalent to $P^{(3,j)}_{\mathsf{hyb}}$ on the punctured input. Since the two programs also agree on all other inputs, the two programs are therefore functionally equivalent and the claim follows from the security of $i\mathcal{O}$ against a non-uniform distinguishing adversary that is given the preprocessing as non-uniform advice. □

– *Hybrid* $\mathcal{H}_{5,j}$. We define $\mathcal{H}_{5,j}$ to be the hybrid distribution where we replace the obfuscation of the program $P^{(4,j)}_{\mathsf{hyb}}$ with an obfuscation of the program $P^{(5,j)}_{\mathsf{hyb}}$. The program $P^{(5,j)}_{\mathsf{hyb}}$ is described in Hybrid 5 and has the hardcoded master PPRF key K_0 replaced with a punctured PPRF key $K^*_0 \leftarrow \mathsf{PPRF.Puncture}(K_0, (c_j \| f_j \| i_j))$.

Hybrid 5: (Parameterized by $j \in \{0, 1, \ldots, 2^n\}$ and a universal circuit $\mathcal{U}$)

Hardcoded: $(\mathsf{hk}, K^*_0, K^*, d_j \oplus R, y, (c_j \| f_j \| i_j))$.
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**

if $(c_{\widehat{X}}, f, i) < (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K^*_0, c_{\widehat{X}} \| f \| i)$
2: **return** R_i

if $(c_{\widehat{X}}, f, i) = (c_j \| f_j \| i_j)$
1: **return** $d_j \oplus R$

if $(c_{\widehat{X}} \| f \| i) > (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K^*, c_{\widehat{X}} \| f \| i)$
2: $d := \mathcal{U}(f, x_i, y)$
3: **return** $d \oplus R_i$

2: **else return** $\perp$

Claim. $\mathcal{H}_{5,j} \approx_c \mathcal{H}_{4,j}$ assuming the security of $i\mathcal{O}$.

Proof. The master PPRF key K_0 is never used to evaluate the punctured input in $\mathcal{H}_{4,j}$ and so we can conclude the two programs are functionally equivalent. The claim follows from the security of $i\mathcal{O}$ against non-uniform distinguishing adversaries (we still require the preprocessing to compute $d_j \oplus R$ as non-uniform advice). □

– *Hybrid* $\mathcal{H}_{6,j}$. We define $\mathcal{H}_{6,j}$ to be the hybrid distribution where we replace the obfuscation of the program $P^{(5,j)}_{\mathsf{hyb}}$ with an obfuscation of the program $P^{(6,j)}_{\mathsf{hyb}}$. The program $P^{(6,j)}_{\mathsf{hyb}}$ is described in Hybrid 6 and has the hardcoded output $d_j \oplus R$ replaced with the value $R^* := \mathsf{PPRF.Eval}(K_0, c_j \| f_j \| i_j)$, computed using the PPRF master key K_0.

Hybrid 6: (Parameterized by $j \in \{0, 1, \ldots, 2^n\}$ and a universal circuit $\mathcal{U}$)

Hardcoded: $(\mathsf{hk}, K^*_0, K^*, R^*, y, (c_j \| f_j \| i_j))$.
Input: $(c_{\widehat{X}}, x_i, (\widehat{x}_i, r_i), i, \pi_i, f)$.
Procedure:

1: **if** $\widehat{x}_i = \mathsf{Commit}(x_i; r_i) \wedge \mathsf{SSB.Verify}(\mathsf{hk}, c_{\widehat{X}}, \widehat{x}_i, i, \pi_i) = 1$ **then**

if $(c_{\widehat{X}}, f, i) < (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K^*_0, c_{\widehat{X}}, f, i)$
2: **return** R_i

if $(c_{\widehat{X}}, f, i) = (c_j \| f_j \| i_j)$
1: **return** R^*

if $(c_{\widehat{X}} \| f \| i) > (c_j \| f_j \| i_j)$
1: $R_i := \mathsf{PPRF.Eval}(K^*, c_{\widehat{X}} \| f \| i)$
2: $d := \mathcal{U}(f, x_i, y)$
3: **return** $d \oplus R_i$

2: **else return** $\perp$

Claim. $\mathcal{H}_{6,j} \approx_c \mathcal{H}_{5,j}$ assuming the security of the PPRF.

Proof. The proof is almost identical to the proof for $\mathcal{H}_{3,j} \approx_c \mathcal{H}_{2,j}$. In particular, observe that $d_j \oplus R$ is distributed as a uniformly random string, by the fact that R is uniformly random. As such, indistinguishability is implied by the puncturing security of the PPRF. □

– *Hybrid* $\mathcal{H}_{7,j}$. We define $\mathcal{H}_{7,j}$ to be the hybrid distribution where we set $\mathsf{hk} \leftarrow \mathsf{SSB.Gen}(1^\lambda, L, i_{j+1})$. That is, we make the SSB hash statistically binding on index i_j, as parsed from the $(j+1)$-st canonical input $(c_{j+1} \| f_{j+1} \| i_{j+1})$.

Claim. $\mathcal{H}_{7,j} \approx_c \mathcal{H}_{6,j}$ assuming the index-hiding of the SSB hash function.

Proof. On the one hand, if the switch from $\mathcal{H}_{6,j}$ to $\mathcal{H}_{7,j}$ has $i_j = i_{j+1}$ (recall that we are switching over from one canonical *input* to the next, which may not change the value of i_j and i_{j+1}), the claim follows trivially. On the other hand, if the switch from $\mathcal{H}_{6,j}$ to $\mathcal{H}_{7,j}$ has $i_j \neq i_{j+1}$, then the claim follows directly from the index-hiding property of the SSB hash. □

Claim. $\mathcal{H}_{1,j+1} \approx_c \mathcal{H}_{7,j}$ assuming the security of $i\mathcal{O}$.

Proof. The claim follows immediately by noticing that the two programs compute identical values at the punctured input, making them functionally equivalent. The claim then follows directly from the security of $i\mathcal{O}$. □

Lemma 8. *$\mathcal{H}_{1,2^n} \approx_c \mathcal{H}_{1,0}$ assuming the sub-exponential security of $i\mathcal{O}$, the existence of sub-exponentially secure one-way functions, the existence of injective one-way functions (for perfectly-binding commitments), and the security of somewhere statistical binding hash functions with (perfect) binding.*

Proof. First, following Section 6.3, we can complexity leverage by assuming sub-exponential security of $i\mathcal{O}$ and one-way functions. Then, we have that $\mathcal{H}_{1,j+1} \approx_c \mathcal{H}_{7,j}$ and $\mathcal{H}_{7,j} \approx_c \mathcal{H}_{1,j}$, which implies that $\mathcal{H}_{1,j} \approx_c \mathcal{H}_{1,j+1}$. It then follows from Section 6.3 that we can set parameters such that there does not exist an efficient distinguisher $\mathcal{D}$ with a sub-exponential distinguishing advantage between hybrids $\mathcal{H}_{1,0}$ and $\mathcal{H}_{1,2^n}$.

Finally, we note that injective one-way functions give us perfectly binding commitment schemes (cf. Definition 17). This concludes the proof of the lemma. □

Corollary 1. *$\mathcal{S}(\mathsf{crs}) \approx_c \mathcal{H}_{1,2^n}$ assuming the security of $i\mathcal{O}$.*

Proof. Indistinguishability follows from the fact that the program $P^{(1,2^n)}_{\mathsf{hyb}}$ (cf. Hybrid 1) does not use the hardcoded input y at all which, by the security of $i\mathcal{O}$, makes the obfuscation of $P^{(1,2^n)}_{\mathsf{hyb}}$ computationally indistinguishable to the obfuscation of P^{sim}. This concludes the proof of the corollary. □

This concludes the proof of Proposition 3. ■

Remark 14 (On the security of the SSB hash). Note that though the above proof relies on 2^n hybrids, the number of times that index hiding security of SSB hashing is invoked is only L (which is the batch size). Therefore, it is sufficient to rely on polynomially-secure SSB hashing rather than a sub-exponential secure version.

Remark 15 (Is perfect binding required?). We required using a commitment scheme and SSB hash with *perfect* binding (as we defined in Definition 15). The reason is that, with an exponential number of hybrids in the digest length $n = n(\lambda_{\mathsf{ssb}}) \in \mathsf{poly}(\lambda_{\mathsf{ssb}})$, the existence of a negligible fraction of *non-binding* digests is already sufficient to invalidate the proof of the claim showing $\mathcal{H}_{4,j} \approx_c \mathcal{H}_{3,j}$.