On the Communication Complexity of Secure Function Evaluation with Long Output

Pavel Hubáček* Daniel Wichs†

Abstract

We study the communication complexity of secure function evaluation (SFE). Consider a setting where Alice has a short input x_A, Bob has an input x_B and we want Bob to learn some function $y = f(x_A, x_B)$ with large output size. For example, Alice has a small secret decryption key, Bob has a large encrypted database and we want Bob to learn the decrypted data without learning anything else about Alice's key. In a trivial insecure protocol, Alice can just send her short input x_A to Bob. However, all known SFE protocols have communication complexity that scales with size of the output y, which can potentially be much larger. Is such "output-size dependence" inherent in SFE?

Surprisingly, we show that output-size dependence can be avoided in the *honest-but-curious* setting. In particular, using indistinguishability obfuscation (iO) and fully homomorphic encryption (FHE), we construct the first honest-but-curious SFE protocol whose communication complexity only scales with that of the best insecure protocol for evaluating the desired function, independent of the output size. Our construction relies on a novel way of using iO via a new tool that we call a "somewhere statistically binding (SSB) hash", and which may be of independent interest.

On the negative side, we show that output-size dependence is inherent in the *fully malicious* setting, or even already in an *honest-but-deterministic* setting, where the corrupted party follows the protocol as specified but fixes its random tape to some deterministic value. Moreover, we show that even in an offline/online protocol, the communication of the online phase must have output-size dependence. This negative result uses an incompressibility argument and it generalizes several recent lower bounds for functional encryption and (reusable) garbled circuits, which follow as simple corollaries of our general theorem.

1 Introduction

We study the communication complexity of secure function evaluation (SFE). For simplicity, we focus on the case of a two-party functionality $y = f(x_A, x_B)$, where Alice and Bob start with inputs x_A, x_B respectively and we want Bob to learn the output y. Alice should not learn anything about Bob's input x_B or the output y, and Bob should not learn anything about Alice's input x_A *beyond* learning the output $y = f(x_A, x_B)$.

Traditional approaches to SFE, for example based on Yao garbled circuits [Yao82a], have communication complexity which is proportional to the circuit size of the function f. The breakthrough results on fully-homomorphic encryption (FHE) by Gentry [Gen09] and follow-up works (e.g., [BV11; BGV12; GSW13]) gave the first general SFE solutions whose communication complexity is independent of the circuit size of f, and only scales with the input and output size of f.[1]

*Aarhus University. E-mail: hubacek@cs.au.dk. Research supported by ERC Starting Grant 279447 and the CFEM and CTIC research centers. Part of this work was done while visiting the Northeastern University.

†Northeastern University. E-mail: wichs@ccs.neu.edu. Research supported by NSF grants 1347350, 1314722.

[1] Prior to FHE, it was known how to beat the circuit-size barrier in many interesting cases, but not in general. See for example Naor and Nissim [NN01].

Communication Complexity of SFE using FHE. Using FHE, we can get SFE solutions whose communication complexity only scales with the *input* and *output* size of the function f being computed. For example, in the honest-but-curious setting, we get a solution where Bob encrypts his input x_B to Alice via a compact and circuit-private FHE for which he knows the secret key, Alice then runs the computation homomorphically on the received ciphertext and her input x_A to get an encryption of y which she sends back to Bob, and Bob decrypts and learns y. This achieves communication complexity that scales with $|x_B| + |y|$. Alternatively, we can get a similar protocol with communication complexity that scales with $|x_A| + |y|$.

More generally, we can take any *insecure* protocol evaluating a function f with low communication complexity and convert it into a secure protocol as follows. Alice and Bob first run a secure distributed key-generation protocol for an FHE scheme, which gives them a common public-key pk and secret-shares sk_A, sk_B of the secret key $sk = sk_A \oplus sk_B$. They then encrypt their inputs under pk and execute the insecure protocol for evaluating f, by running it homomorphically under the FHE scheme, so that Bob eventually learns an encryption of the output y. Alice and Bob can then run a secure distributed decryption procedure using their shares sk_A, sk_B and the encryption of y so that Bob learns the decrypted output y (but nothing else). The distributed key-generation and decryption protocols can be implemented generically using an arbitrary SFE scheme.[2] If $\mathbf{CC}(f)$ is the best communication complexity for an efficient protocol evaluating f *without* any security requirements and λ is the security parameter, then the above approach yields an SFE with communication complexity $\mathsf{poly}(\lambda)(\mathbf{CC}(f) + |y|)$ where the $\mathsf{poly}(\lambda)$ term is some fixed polynomial independent of the function f. Moreover, using succinct zero-knowledge arguments of Kilian [Kil92], we can even make the above protocols secure in the fully malicious setting without asymptotically increasing the communication complexity.

However, in all known SFE protocols, including the above-described solutions, the communication complexity of the protocol exceeds the output size $|y|$ of the computation. We say that such protocols have *"output-size dependence"*. As the main question of the paper, we ask if output-size dependence is inherent in SFE.

Is output-size dependence inherent in SFE? If we didn't require any security, then there is a trivial protocol where Alice just sends her input x_A to Bob who computes $y = f(x_A, x_B)$ himself, with communication $|x_A|$ independent of the output size $|y|$. Can we achieve this type of efficiency while maintaining security? For example, imagine that Alice has a short seed x for a pseudorandom generator (PRG) G, Bob does not have any input, and we want Bob to learn a huge PRG output $y = G(x)$. Can we do this with communication complexity which only depends on $|x|$ but not on $|y|$? Alternatively, imagine Alice has a small secret decryption key sk, Bob has a large encrypted database $c = \mathsf{Enc}_{pk}(\mathsf{DB})$ and we want Bob to learn the plaintext database $\mathsf{DB} = \mathsf{Dec}_{sk}(c)$ without learning anything else about sk. Can we do this with communication complexity independent of $|\mathsf{DB}|$? In all of these cases, we would like to have an SFE protocol that avoids output-size dependence.

Our Results. On the negative side, we show that output-size dependence is inherent for SFE in the *fully malicious* setting. In fact, it is already required even in an *honest-but-deterministic* setting, where the corrupted party follows the protocol as specified but fixes its random tape to some deterministic value (say, all 0s). More specifically, we show that in any honest-but-deterministic SFE scheme, the communication from Alice to Bob must exceed the "Yao incompressibility entropy" of Bob's output conditioned on his input, and in general, this can be as large as Bob's output size. Moreover, we extend this result to protocols in the offline/online setting,

[2] Alternatively, there are much simpler and more efficient distributed key-generation and decryption procedures using the algebraic structure of specific FHE schemes. See for example [AJL+12].

where the parties can run an offline phase before knowing their inputs.[3] We show that, no matter how much communication takes place in the offline phase, the communication of the online phase must still satisfy output-size dependence. This negative result uses an "incompressibility argument" which has been used in several recent works giving negative results and/or lower bounds for functional encryption and garbled circuits [AIKW13; AGVW13; DIJ+13; GGJS13; GHRW14]. Our main contribution is to give a (relatively straight-forward) generalization of this technique to proving lower bounds on the communication complexity of general SFE, and then show that all of the prior uses of this technique follow as simple corollaries of our general theorem.

On the positive side, we show that output-size dependence can surprisingly be avoided in the *honest-but-curious* setting. In particular, using indistinguishability obfuscation (iO) and fully-homomorphic encryption (FHE), we construct the first general honest-but-curious SFE protocols that avoid output-size dependence. We give two such protocols for evaluating an arbitrary function f where Bob learns the output. The first protocol achieves communication $\mathsf{poly}(\lambda) + |x_A|$, where x_A is Alice's input. The second protocol achieves communication $\mathsf{poly}(\lambda)\mathbf{CC}(f)$ which only scales with the communication-complexity $\mathbf{CC}(f)$ of the most efficient protocol for evaluating f without any security. In both cases, the $\mathsf{poly}(\lambda)$ term is some fixed polynomial in the security parameter λ, independent of the function f. Since there is always a simple insecure protocol where Alice sends her input to Bob, we have $\mathbf{CC}(f) \leq |x_A|$, and there are functions for which the gap is large $\mathbf{CC}(f) \ll |x_A|$. Therefore the latter protocol may be better in some instances, although it incurs a multiplicative rather than additive overhead in the security parameter. Using either of these protocols, we get an SFE solution to the problem where Bob has a large encrypted database and wants to securely learn the decryption under a short key held by Alice, using communication complexity which is independent of the database size.

1.1 Our Techniques: Negative Result

Let us begin by describing the technique behind the negative result. For concreteness, consider a two-party computation protocol where Alice has a secret key k for a pseudorandom function (PRF) $f_k : \mathbb{N} \to \{0,1\}$, Bob does not have any input, and we want Bob to learn the PRF outputs $y_1 = f_k(1), \ldots, y_L = f_k(L)$ for some large integer L. For contradiction, assume that we had an SFE protocol for this task where the communication complexity from Alice to Bob is $L' < L$ bits. Let's look at the case where Alice is honest and has a uniformly random key k as her input, while a corrupted Bob uses an "honest-but-deterministic" strategy, meaning that he follows the specified protocol but fixes his random tape to some deterministic value (say, all 0s). The security of the SFE protocol implies that there is a simulator which gets Bob's output $y_1 = f_k(1), \ldots, y_L = f_k(L)$ and must simulate the view of Bob, denoted $\mathsf{view}_{\mathrm{Bob}}$. The simulated $\mathsf{view}_{\mathrm{Bob}}$ consists of messages from Alice to Bob (of size L') that cause Bob to output $y_1, \ldots, y_L$. But this means that the simulator can *efficiently compress* the outputs $y_1, \ldots, y_L$ into a shorter string $\mathsf{view}_{\mathrm{Bob}}$ of size $L' < L$ bits from which we can efficiently recover the output (by running Bob's protocol with the fixed randomness). This in turn contradicts the fact that the outputs are pseudorandom and therefore incompressible, showing that such an SFE cannot exist.

As mentioned, our actual result generalize the above example in several ways. Firstly, we show that the communication from Alice to Bob in any SFE for any function f must exceed the "Yao incompressibility entropy" [Yao82b; HLR07] of Bob's output given Bob's input. In the above example, the incompressibility entropy of the output is L bits. Secondly, we extend this result to protocols in the offline/online setting, where we show that the same lower-bound

[3] We require the offline phase to be simulatable on its own, prior to the online inputs being specified.

applies to the online phase, no matter how much communication takes place in the offline phase. In this setting, we require that the simulator can simulate the offline phase before the online inputs are chosen, and therefore without knowing the output of the computation.

Lastly, we show that the above negative result implies many prior lower-bounds on functional encryption and/or (reusable) garbled circuits. In particular, all of these results follow by showing that the desired primitive would immediately yield an SFE protocol (possibly in the offline/online setting) whose communication complexity would beat our lower bound.

1.2 Our Techniques: Positive Result

Surprisingly, we show that the negative result can be avoided in the honest-but-curious setting.[4] Before we describe our solution, it is instructive to see where the negative result fails. In the negative result, we relied on the fact that the simulated view of an "honest-but-deterministic" Bob, denoted $\mathsf{view}_{\mathrm{Bob}}$, can be used to reconstruct the output of the function, which consists of pseudorandom values $y_1 = f_k(1), \ldots, y_L = f_k(L)$. Since this view only contained the communication from Alice to Bob, which we assumed to be shorter than the output size L, this served as a compression of the output leading to a contradiction. In the "honest-but-curious" setting, the view of Bob also contains all of his random coins used during the protocol execution, which may be arbitrarily long. Therefore, in this setting, $\mathsf{view}_{\mathrm{Bob}}$ is no longer compressing even if the communication complexity is small, and the negative result fails.

At first it may appear that the above observation cannot help us. In the real protocol execution, Bob's coins are truly random and independent of the output; how can we use the random coins to represent/reconstruct the output $y_1, \ldots, y_L$? We rely on the fact that the simulator can choose the "simulated random coins" for Bob in a way that is not truly random, and in fact can somehow embed into the random coins information about the outputs $y_1, \ldots, y_L$, while still making the coins appear random to a distinguisher.

We now describe our positive result in several steps, focusing on the above example of PRF evaluation for concreteness. This high-level overview doesn't match the actual constructions in the paper, but it elucidates the main ideas.

First Attempt: Just Obfuscate. As a first attempt, consider a protocol where Alice constructs a small circuit $C_k(i)$ with a hard-coded key k which gets as input an index $i \in \{1, \ldots, L\}$ and outputs $f_k(i)$. Alice obfuscates this circuit and sends it to Bob who then evaluates it on the values $1, \ldots, L$ to get his output. Since the size of the circuit is independent of L (ignoring logarithmic factors), so is the communication complexity of this protocol. On the positive side, the use of obfuscation might already hide some information about Alice's input k. On the negative side, there is no hope of simulating this protocol. Indeed, since Bob does not send any communication to Alice, there is no difference between proving the security of this protocol for an honest-but-curious Bob vs. a fully malicious Bob, and our negative result rules this out.

Second Attempt: Hash Randomness then Obfuscate. Our main idea for overcoming the negative result is to incorporate the random coins of Bob into the protocol. Let's consider the following modification. Bob first chooses L random bits $r_1, \ldots, r_L$ and hashes them using a Merkle Tree to derive $z = H(r_1, \ldots, r_L)$. A Merkle Tree has the property that Bob can efficiently "open" any bit r_i of the pre-image by providing a short opening π_i. Bob sends the hash z to Alice. Alice now constructs a small circuit $C_{k,z}(i, r_i, \pi_i)$ that has k, z hard-coded, gets

[4] One surprising aspect of this result is that Bob's randomness is needed for Alice's security! We are not aware of any previous protocol which would provide security for Alice when Bob is honest-but-curious yet would be insecure if Bob is honest-but-deterministic.

as input $i \in \{1, \ldots, L\}, r_i \in \{0,1\}$ and a short opening π_i, verifies that π_i is a valid opening of the i'th pre-image bit to r_i, and if so outputs $f_k(i)$ (if not outputs 0). Alice obfuscates this circuit $C_{k,z}$ and sends it to Bob, who then evaluates it on the values $i \in \{1, \ldots, L\}$ by also providing the correct bit r_i and the opening π_i for each evaluation. The size of the circuit $C_{k,z}$ is independent of L and therefore so is the communication complexity of this protocol.

Notice that Bob commits himself ahead of time to providing some random bits r_i to the obfuscated circuit on each evaluation. The circuit checks that it gets the right bit, but then essentially ignores it afterwards. The main idea of the simulation strategy is to choose the random coins r_i on behalf of Bob in a way that embeds information about the outputs y_i and to change the circuit being obfuscated so that it uses the inputs r_i to compute the output without knowing k. The simulator chooses his own PRF key k' (unrelated to Alice's key k which the simulator does not know) and sets the simulated random coins to $r_i := f_{k'}(i) \oplus y_i$. Then it creates an obfuscation of the circuit $C'_{k',z}(i, r_i, \pi_i)$ which checks the opening π_i as before, and if the opening is valid, it now outputs $r_i \oplus f_{k'}(i)$ instead of $f_k(i)$. In both cases, if the circuits $C_{k,z}$ and $C'_{k',z}$ get the correct inputs (i, r_i, π_i) they produce the same outputs y_i.

The indistinguishability of simulation boils down to showing that one cannot distinguish an obfuscation of $C_{k,z}$ and $C'_{k',z}$ even given $r_1, \ldots, r_L$, k and k', where $z = H(r_1, \ldots, r_L)$. Functionally, these circuits only differ on inputs of the form (i, r'_i, π'_i) where $r'_i \neq r_i$ differs from the bit that was hashed to create z and π'_i is a valid opening. By the security of the Merkle Tree, such inputs are hard to find. Therefore, we could already show the security of the above construction by relying on differing-inputs obfuscation (diO) [BGI+01; ABG+13; BCP14]. However, diO is a strong assumption and there is some evidence that it may not hold in general [GGHW14]. Therefore, we would like a solution based on the weaker and better studied notion of indistinguishability obfuscation (iO) [BGI+01; GGH+13].[5]

Final Attempt: Hash Carefully and Use iO. It turns out that we can also prove the security of the above construction using only iO, by being more careful and relying on special type of hash function. We believe that this technique may have broader applicability.

Abstracting out the above construction, we have two circuits C, C' which have a hard-coded hash-output $z = H(r_1, \ldots, r_L)$ and they only differ on inputs of the type (i, r'_i, π'_i) where $r'_i \neq r_i$ differs from the hashed bit in position i, and π'_i is a valid opening for r'_i. Since the hash is compressing, such inputs necessarily exist. However, we'd like to rely on iO to show that obfuscations of C and C' are indistinguishable. We show that we can do this if the hash function satisfies a new notion of security which we call a *"somewhere statistically binding" (SSB) hash.*

An SSB hash H_{hk} has a short public "hashing key" hk. Just like in a Merkle Tree, we can take $z = H_{\mathsf{hk}}(r_1, \ldots, r_L)$ and, for any position i, produce a short opening π_i to certify r_i as the correct bit of the pre-image in that position. Moreover, there is now a method of choosing the hash key hk with a special "binding index" i^* and we require the following two security properties:

- $z = H_{\mathsf{hk}}(r_1, \ldots, r_L)$ is statistically binding for position i^*, meaning that there does not exist a valid opening π'_{i^*} that would open position i^* to the wrong bit $r'_{i^*} \neq r_{i^*}$.
- The hash key hk does not reveal anything about which index i^* is the binding index.

We show how to construct such SSB hash functions by combining the idea of a Merkle Tree with fully homomorphic encryption. We believe that this primitive may find other applications.

[5] In both notions of obfuscation, we want to ensure that the obfuscations of two different circuits C, C' are indistinguishable. For iO, we assume that C, C' agree on all inputs, whereas for diO we only assume that it is computationally infeasible to find an input on which they disagree.

Using an SSB hash, we show that obfuscations of C and C' are indistinguishable via a careful hybrid argument. We can define hybrid circuits $C_{i^*}(i, r_i, \pi_i)$ that evaluate C' when $i \leq i^*$ and C otherwise, so that $C_0 = C$ and $C_L = C'$. For $i^* = 1, \ldots, L$ we define a series of hybrid distributions where we first change the way we choose hk to be binding on index i^* and then we change the circuit being obfuscated to C_{i^*}. Each time we change the circuit being obfuscated from C_{i^*-1} to C_{i^*}, the two circuits being considered are functionally equivalent since the SSB hash is binding on index i^*. Therefore, we get a proof of security using only iO rather than diO.

General Result. So far, we described a specific example for our positive result where we avoid output-size dependence in the concrete case of PRF evaluation. However, the above ideas generalize to providing a general honest-but-curious two-party SFE protocol for any function f, so as to achieve the positive results we described previously.

Can Obfuscation be Avoided? We do not know if iO can be avoided in the above positive result but, in Section 3.4, we present some evidence that at least a weak flavor of obfuscation is inherent. It remains an interesting problem to explore this further and to see what are the minimal assumptions under which we can avoid "output-size dependence" in the honest-but-curious setting.

2 Definitions for SFE

Let $f : \{0,1\}^{\ell_A(\lambda)} \times \{0,1\}^{\ell_B(\lambda)} \to \{0,1\}^{L(\lambda)}$ be a function family. We consider the *secure function evaluation problem* with two parties Alice and Bob, where Alice has a private input $x_A \in \{0,1\}^{\ell_A}$, Bob has a private input $x_B \in \{0,1\}^{\ell_B}$, and Bob wishes to obtain the evaluation $y = f(x_A, x_B)$.

Honest-But-Curious SFE. For our positive result, we will rely on the notion of honest-but-curious adversaries, where the adversarial party is assumed to follow the protocol specification completely and hopes to learn some unintended information. For an *honest-but-curious* Bob, we define a random variable denoting his view of the protocol by $\mathsf{view}_B^{\Pi}(x_A, x_B, \lambda) = (x_B, r_B, m_1, \ldots, m_t)$ where r_B are the random coins used by Bob, and $m_1, \ldots, m_t$ are the messages from Alice to Bob. We define the view of an honest-but-curious Alice, $\mathsf{view}_A^{\Pi}(x_A, x_B, \lambda)$, analogously.

Definition 2.1 (Honest-But-Curious SFE)**.** *We say that two-party protocol* Π securely evaluates $f : \{0,1\}^{\ell_A} \times \{0,1\}^{\ell_B} \to \{0,1\}^L$ in the presence of honest-but-curious adversaries*, if there exists a* PPT *simulator* $S = (S_A, S_B)$ *such that for all* $x_A \in \{0,1\}^{\ell_A}$ *and* $x_B \in \{0,1\}^{\ell_B}$ *it holds that*

$$\{\mathsf{sim}_{A,\lambda}\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{\mathsf{view}_A^{\Pi}(x_A, x_B, \lambda)\}_{\lambda \in \mathbb{N}} , \tag{1}$$

$$\{\mathsf{sim}_{B,\lambda}\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{\mathsf{view}_B^{\Pi}(x_A, x_B, \lambda)\}_{\lambda \in \mathbb{N}} , \tag{2}$$

where $\mathsf{sim}_{A,\lambda} \leftarrow S_A(1^\lambda, x_A)$ *and* $\mathsf{sim}_{B,\lambda} \leftarrow S_B(1^\lambda, x_B, f(x_A, x_B))$.

We sometimes consider "one-sided" security against honest-but-curious Bob, in which case we only require (2) *to hold.*

Honest-But-Deterministic SFE. For our negative results, we will rely on a notion of *honest-but-deterministic* adversaries, where the adversarial party follows the protocol specification, *except* that it refuses to choose truly random coins, and instead sets its random tape to some fixed/deterministic value – say, the all 0 string. This adversarial model is much weaker

than the fully malicious one, making our negative results stronger. We will also only consider one-sided security against an honest-but-deterministic Bob, but not require any security against an adversarial Alice. Again, this weakening of the security model makes our results stronger. Lastly, we consider offline/online protocols $\Pi = (\Pi^{\text{off}}, \Pi^{\text{on}})$, comprising of two phases:

- The *offline phase* protocol Π^{off} is run independently of the inputs x_A, x_B and it allows the parties to do some pre-processing. Both parties receive their respective inputs only after the end of the offline phase.
- The *online phase* protocol Π^{on} is run by the parties using their inputs x_A, x_B and any state retained from the offline phase. It results in Bob outputting $y = f(x_A, x_B)$.

We can think of standard SFE protocols as only containing an online phase. We will give a lower-bound on the communication complexity of the online phase in an offline/online protocol, no matter how much communication takes place in the online phase.

The execution of a protocol $\Pi = (\Pi^{\text{off}}, \Pi^{\text{on}})$ between Alice and honest-but-deterministic Bob defines a random variable for Bob's view of the protocol:

$$(\mathsf{detview}_B^{\Pi^{\text{off}}}, \mathsf{detview}_B^{\Pi^{\text{on}}})(x_A, x_B, \lambda) = ((m_1^{\text{off}}, \ldots, m_s^{\text{off}}), (x_B, m_1^{\text{on}}, \ldots, m_t^{\text{on}}))$$

where the offline part consists of the protocol messages from Alice to Bob in the offline phase, and the online part consists of Bob's input and the protocol messages from Alice to Bob in the online phase. The protocol messages chosen by Alice follow the protocol with true randomness, while those from Bob follow the protocol with the random tape set to the all 0s string.[6]

Definition 2.2 (Offline/Online SFE Secure against Honest-But-Deterministic Bob)**.** *We say that an offline/online two-party protocol* $\Pi = (\Pi^{\text{off}}, \Pi^{\text{on}})$ evaluates $f : \{0,1\}^{\ell_A} \times \{0,1\}^{\ell_B} \to \{0,1\}^L$ with security against honest-but-deterministic Bob*, if there exists a simulator* $S = (S^{\text{off}}, S^{\text{on}})$ *such that for all* $x_A \in \{0,1\}^{\ell_A}$ *and* $x_B \in \{0,1\}^{\ell_B}$ *it holds that*

$$\{\mathsf{sim}_{B,\lambda}^{\text{off}}, \mathsf{sim}_{B,\lambda}^{\text{on}}\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \{(\mathsf{detview}_B^{\Pi^{\text{off}}}, \mathsf{detview}_B^{\Pi^{\text{on}}})(x_A, x_B, \lambda)\}_{\lambda \in \mathbb{N}} ,$$

where $(\mathsf{sim}_{B,\lambda}^{\text{off}}, \mathsf{state}) \leftarrow S^{\text{off}}(1^\lambda)$ *and* $\mathsf{sim}_{B,\lambda}^{\text{on}} \leftarrow S^{\text{on}}(x_B, f(x_A, x_B), \mathsf{state})$.

A crucial but subtle aspect of the above definition for offline/online SFE is that the simulator S^{off} must simulate the offline phase without knowing Bob's input x_B or the output $f(x_A, x_B)$. For example, this is required if the inputs can be chosen (e.g., by the adversary/environment) adaptively after the offline phase. In fact, our lower bound in Section 4.2 can be overcome if the simulator is allowed to know the output when simulating the offline part. Indeed Yao garbled circuits (discussed further in Section 4.4) give such an offline/online protocol with low communication complexity in the online part if the offline simulator is given the output.

3 Positive Results in the Honest-But-Curious Setting

We now describe our positive results, giving general SFE protocols in the honest-but-curious setting, whose communication-complexity is independent of the output size of the function being computed. As explained in the introduction, we will rely on a new type of security for hash functions and we begin by describing this new primitive.

[6] Our results would hold even if we modified the definition so that an honest-but-deterministic Bob uses true randomness in the offline phase but is deterministic in the online phase. We choose to omit this for simplicity.

3.1 Somewhere Statistically Binding Hash Functions

Definition. A *somewhere statistically binding (SSB) hash* function allows us to create a *short* hash $y = H_{\mathsf{hk}}(x)$ of some *long* value $x = (x[0], \ldots, x[L-1])$ and later efficiently "prove" that the i'th block of x takes on some particular value $x[i] = u$ by providing a *short* opening π. The size of the hash $y = H_{\mathsf{hk}}(x)$, the size of the opening π and the time to verify π should be bounded by some fixed polynomials in the security parameter and unrelated to the potentially huge size of x. So far, this problem can be solved using Merkle Trees, where such an opening consists of the hash values of all the sibling nodes along the path from the root of the tree to the i'th leaf. However, the definition of SSB hash has an additional statistical requirement: it allows us to choose the *hashing key* hk with respect to some special "binding index" i in such a way that the hash $y = H_{\mathsf{hk}}(x)$ is statistically binding on the i'th block of the input, meaning that there only exists a valid opening for a single choice of $x[i]$. The index i on which the hash is statistically binding should remain hidden by the hashing key hk. The formal definition is below.

Definition 3.1 (SSB Hash)**.** *A somewhere statistically binding (SSB) hash consists of* PPT *algorithms* $(\mathsf{Gen}, H, \mathsf{Open}, \mathsf{Verify})$ *along with a block alphabet* $\Sigma = \{0,1\}^{\ell_{blk}}$*, output size* ℓ_{hash} *and opening size* ℓ_{opn}*, where* $\ell_{blk}(\lambda), \ell_{hash}(\lambda), \ell_{opn}(\lambda)$ *are some fixed polynomials in the security parameter. The algorithms have the following syntax:*

- $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, i)$ *takes as input an integer* $L \leq 2^\lambda$ *and index* $i \in \{0, \ldots, L-1\}$ *(both of these are in binary) and outputs a public hashing key* hk*.*
- $H_{\mathsf{hk}} : \Sigma^L \to \{0,1\}^{\ell_{hash}}$ *is a deterministic polynomial time algorithm that takes as input* $x = (x[0], \ldots, x[L-1]) \in \Sigma^L$ *and outputs* $H_{\mathsf{hk}}(x) \in \{0,1\}^{\ell_{hash}}$*.*
- $\pi \leftarrow \mathsf{Open}(\mathsf{hk}, x, j)$*: Given the hash key* hk*,* $x \in \Sigma^L$ *and an index* $j \in \{0, \ldots, L-1\}$*, creates an opening* $\pi \in \{0,1\}^{\ell_{opn}}$*.*
- $\mathsf{Verify}(\mathsf{hk}, y, j, u, \pi)$*: Given a hash key* hk *and* $y \in \{0,1\}^{\ell_{hash}}$*, an integer index* $j \in \{0, \ldots, L-1\}$*, a value* $u \in \Sigma$ *and an opening* $\pi \in \{0,1\}^{\ell_{opn}}$*, outputs a decision* $\in \{\mathsf{accept}, \mathsf{reject}\}$*. This is intended to verify that a pre-image* x *of* $y = H_{\mathsf{hk}}(x)$ *has* $x[j] = u$*.*

We require the following properties:

Correctness: *For any integers* $L \leq 2^\lambda$ *and* $i, j \in \{0, \ldots, L-1\}$*, any* $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, i)$*,* $x \in \Sigma^L$*,* $\pi \leftarrow \mathsf{Open}(\mathsf{hk}, x, j)$*: we have* $\mathsf{Verify}(\mathsf{hk}, H_{\mathsf{hk}}(x), j, x[j], \pi) = \mathsf{accept}$*.*

Index Hiding: *We consider the following game between an attacker* $\mathcal{A}$ *and a challenger:*

- *The attacker* $\mathcal{A}(1^\lambda)$ *chooses an integer* L *and two indices* $i_0, i_1 \in \{0, \ldots, L-1\}$*.*
- *The challenger chooses a bit* $b \leftarrow \{0,1\}$ *and sets* $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, i_b)$*.*
- *The attacker* $\mathcal{A}$ *gets* hk *and outputs a bit* b'*.*

We require that for any PPT *attacker* $\mathcal{A}$ *we have* $|\Pr[b = b'] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$ *in the above game.*

Somewhere Statistically Binding: *We say that* hk *is* statistically binding for an *index* i *if there do not exist any values* $y, u \neq u', \pi, \pi'$ *s.t.* $\mathsf{Verify}(\mathsf{hk}, y, i, u, \pi) = \mathsf{Verify}(\mathsf{hk}, y, i, u', \pi') = \mathsf{accept}$*. We require that for any integers* $L \leq 2^\lambda$*,* $i \in \{0, \ldots, L-1\}$ *the key* $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, i)$ *is statistically binding for* i*.*

Remarks. Notice that the output size and the opening size of SSB hash are bounded by some fixed polynomials ℓ_{hash}, ℓ_{opn} and independent of the size of the input x. Also, we note that an SSB hash function is necessarily collision resistant. Intuitively, if an attacker can find $x \neq x'$ such that $H_{\mathsf{hk}}(x) \neq H_{\mathsf{hk}}(x')$then there must be some index i such that $x[i] \neq x'[i]$ and therefore the attacker knows with certainty that the key hk is not binding on index i. This would contradict index hiding. We only make an informal note of this and do not explicitly rely on this property.

Overview of Construction. Our construction combines the ideas behind Merkle Hash Trees with fully homomorphic encryption (FHE, see Appendix B for a formal definition). Let's assume we want to hash some data $x = x[0], \ldots, x[L-1]$ where $L = 2^\alpha$ is a power-of-2. We construct a full binary tree of height α sitting on top of the data x. Each node of the tree is associated with a ciphertext under an FHE scheme. The L leaf nodes are associated with encryptions of $x[0], \ldots, x[L-1]$ respectively, where these ciphertexts are computed deterministically using some fixed random coins. The hashing key $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, i)$ consists of an encrypted path in the tree going to a leaf i. Hashing will consist of homomorphically computing a ciphertext for each node of the tree using the ciphertexts associated with its children and the ciphertexts contained in hk. This is done in a way that ensures that all of the ciphertexts on the path from the root to leaf i contain encryptions of $x[i]$. The output of the hash function is the ciphertext associated with the root of the tree, which is an encryption of $x[i]$ and therefore statistically committing to this value. Analogously to Merkle Trees, opening the hash for some particular block i consist of revealing the ciphertexts associated with all of the siblings along the path from the root to i, which is sufficient to recompute the ciphertext associated with the root. One difficulty is that an adversarial opening may consist of "incorrectly generated ciphertexts" and we cannot guarantee the correctness of homomorphic evaluation over such ciphertexts. Therefore, homomorphic evaluation will only operate on the honestly generated ciphertexts provided as part of the hashing key hk, while the ciphertexts associated with the nodes of the tree will only be used to define the function being evaluated.

Construction. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE scheme. For any polynomial block-size $\ell_{blk} = \ell_{blk}(\lambda)$, we construct an SSB hash function $(\mathsf{Gen}, H, \mathsf{Open}, \mathsf{Verify})$ with alphabet $\Sigma = \{0,1\}^{\ell_{blk}}$ as follows.

- $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, i)$: Assume w.l.o.g. that $L = 2^\alpha$ is a power-of-2 for some integer $\alpha \leq \lambda$. For $j = 0, \ldots, \alpha$: create $(pk_j, sk_j) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Let $(b_\alpha, \ldots, b_1)$ be the binary representation of the index $i \in \{0, \ldots, 2^\alpha - 1\}$.[7] For $j = 1, \ldots, \alpha$, compute $c_j \leftarrow \mathsf{Enc}_{pk_j}((sk_{j-1}, b_j))$. Let $\mathsf{hk} = (pk_0, \ldots, pk_\alpha, c_1, \ldots, c_\alpha)$.
- $y = H_{\mathsf{hk}}(x)$: Let $x = (x[0], \ldots, x[L-1])$. Let T be a binary tree of height α with L leaves. We think of the leaves as being at level 0 and the root of the tree as being at level α. We inductively and deterministically associate a ciphertext ct_v with each vertex $v \in T$. Intuitively, the encrypted bits b_j contained in hk will ensure that the data item $x[i]$ is propagated up the tree in encrypted form. Formally, we define the ciphertexts ct_v inductively as follows:
  - If v is the j'th leaf, we associate to it the ciphertext $ct_v := \mathsf{Enc}_{pk_0}(x[j]; \bar{0})$ to be a deterministically computed encryption of $x[j]$ using fixed randomness $\bar{0}$.
  - Let $v \in T$ be a non-leaf vertex at level $j \in [\alpha]$ with children v_0, v_1 having associated ciphertexts ct_0, ct_1, and let c_j be the ciphertext contained in hk for level j. We

[7] It is useful to think of the bits b_i as tracing out a path in a binary tree going from the root to the leaf i.

associate with v the ciphertext $ct_v = \mathsf{Eval}_{pk_j}(f_{ct_0,ct_1}, c_j)$ where we define the function:

$$f_{ct_0,ct_1}(sk, b) \ : \ \{ \quad \text{Compute: } x_0 = \mathsf{Dec}_{sk}(ct_0), x_1 = \mathsf{Dec}_{sk}(ct_1). \text{ Output } x_b. \ \}$$

Note that the function f_{ct_0,ct_1} is homomorphically evaluated only over the ciphertext $c_j = \mathsf{Enc}_{pk_j}((sk_{j-1}, b_j))$ contained in hk, whereas the ciphertexts ct_0, ct_1 only serve to define the function being evaluated. The output ciphertext ct_v is an encryption under the key pk_j.

The above ensures that for any node v which lies on the path from the root to the leaf at the statistically binding index i, the associated ciphertext ct_v is an encryption of $x[i]$. The output of the hash is the ciphertext ct_v where v is the root of the tree T.

- $\mathsf{Open}(\mathsf{hk}, x, j)$: Perform the computation of $H_{\mathsf{hk}}(x)$ as described above and output the ciphertexts ct_v for each vertex v that's a sibling of some vertex along the path from the root to the leaf at position j.

- $\mathsf{Verify}(\mathsf{hk}, y, j, u, \pi)$: Perform the computation of $H_{\mathsf{hk}}(x)$ as described above using only the provided ciphertexts. In particular, for each vertex v along the path from the root of the tree to leaf j, inductively compute a ciphertext ct_v. In the base case, when v is the j'th leaf, set $ct_v = \mathsf{Enc}_{pk_0}(u; \bar{0})$. Otherwise, if v is not a leaf, than one of its children lies on the path to leaf j in which case the corresponding ciphertext was computed in the previous step, and the sibling ciphertex is provided in the opening π. Therefore, we can compute $ct_v = \mathsf{Eval}_{pk_j}(f_{ct_0,ct_1}, c_j)$ where ct_0, ct_1 are the ciphertexts associated with the children of v and c_j is contained in the hash key hk. Finally, compute the ciphertext ct_v associated with the root of the tree and check that $y \stackrel{?}{=} ct_v$.

Theorem 3.2. *If $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is an FHE scheme then, for any polynomial block-size $\ell_{blk}(\lambda)$, the above construction $(\mathsf{Gen}, H, \mathsf{Open}, \mathsf{Verify})$ is an SSB hash with block-alphabet $\Sigma = \{0,1\}^{\ell_{blk}}$.*

Proof. Correctness of SSB hash follows clearly by observation.

For index hiding security, we rely on the semantic security of the FHE scheme. Let $i_0, i_1 \in \{0, \ldots, L_1\}$ be the two indices chosen by the adversary $\mathcal{A}$ during the course of the index hiding game, with binary representations $i_0 = (b_\alpha, \ldots, b_1)$ and $i_1 = (b'_\alpha, \ldots, b'_1)$ respectively. We define a sequence of indistinguishable hybrids that take us from the challenger using the index i_0 to i_1. First, we define a sequence of α hybrids where we change the ciphertexts contained in the hashing key $\mathsf{hk} = (pk_0, \ldots, pk_\alpha, c_1, \ldots, c_\alpha)$ from $c_j \leftarrow \mathsf{Enc}_{pk_j}((sk_{j-1}, b_j))$ to $c^*_j \leftarrow \mathsf{Enc}_{pk_j}(\bar{0})$ just being an encryption of the all 0 string (starting from $j = \alpha$ and counting down to 1). Then we do another sequence of α hybrids where we change the ciphertexts from $c^*_j \leftarrow \mathsf{Enc}_{pk_j}(\bar{0})$ to $c'_j \leftarrow \mathsf{Enc}_{pk_j}((sk_{j-1}, b'_j))$ (starting from $j = 1$ counting up to α). Each pair of successive hybrids is indistinguishable by the semantic security of the FHE scheme with public key pk_j and the fact that sk_j is not being encrypted in the two hybrids in question.

Finally, for the *somewhere statistically binding* property, we just rely on the correctness of the homomorphic evaluation of the FHE scheme. In particular, assume that $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, i)$ consists of $\mathsf{hk} = (pk_0, \ldots, pk_\alpha, c_1, \ldots, c_\alpha)$ where $(pk_j, sk_j) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $c_j \leftarrow \mathsf{Enc}_{pk_j}((sk_{j-1}, b_j))$ where the binary representation of i is $i = (b_\alpha, \ldots, b_1)$. Assume that $\mathsf{Verify}(\mathsf{hk}, y, i, u, \pi) = \mathsf{accept}$ for some y, u, π. During the verification procedure, for each vertex v_j at level j along the path from the root of the tree to leaf i, we inductively compute a ciphertext ct_{v_j}. We claim, by induction, that each such ciphertext satisfies $\mathsf{Dec}_{sk_j}(ct_{v_j}) = u$. For $j = 0$ (base case), the computation sets $ct_{v_0} := \mathsf{Enc}_{pk_0}(u; \bar{0})$, so the claim holds. For each successive j, the computation sets $ct_{v_j} = \mathsf{Eval}_{pk_j}(f_{ct_0,ct_1}, c_j)$ where $ct_{b_j} = ct_{v_{j-1}}$ lies on the path to leaf i at level $j-1$

while ct_{1-b_j} is provided as part of the opening π and we know nothing about it. By induction $\mathsf{Dec}_{sk_{j-1}}(ct_{b_j}) = u$. By the correctness of homomorphic evaluation, we therefore have

$$\mathsf{Dec}_{sk_j}(ct_{v_j}) = f_{ct_0,ct_1}(sk_{j-1}, b_j) = \mathsf{Dec}_{sk_{j-1}}(ct_{b_j}) = u.$$

This proves the claim. Therefore, the root ciphertext $ct_{v_\alpha} = y$ satisfies $\mathsf{Dec}_{sk_\alpha}(y) = u$. In other word, u is uniquely fixed by y and there cannot exist any $u' \neq u$ and π' such that $\mathsf{Verify}(\mathsf{hk}, y, i, u', \pi') = \mathsf{accept}$. This proves the somewhere statistically binding property.

□

3.2 One-Sided SFE for Multi-Decryption

In the introduction, we gave an example of a functionality where Alice has a short secret key, Bob has a large encrypted database and we want Bob to learn the decryption without learning anything else about Alice's secret key. We now show how to do this in the honest-but-curious setting with communication complexity indpendent of the database size. Then, in the next section, we will leverage this protocol to build general SFE schemes.

Multi-Decryption. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a bit-encryption scheme with ciphertext size $\ell_{ctx} = \ell_{ctx}(\lambda)$. We begin by describing an SFE protocol for the "multi-decryption" functionality where Alice has as input a secret key $sk \in \{0,1\}^{\ell_{sk}}$ and Bob has as input some "database" of L ciphertexts $c_0, \ldots, c_{L-1} \in \{0,1\}^{\ell_{ctx}}$ where L is some polynomial in the security parameter. The functionality gives Bob the decryptions $m_0 = \mathsf{Dec}_{sk}(c_0), \ldots, m_{L-1} = \mathsf{Dec}_{sk}(c_{L-1})$ with $m_i \in \{0,1\}$. In other words, we want an SFE for the function $f^{\mathcal{E},L}_{\mathsf{multi\text{-}dec}}(sk, (c_0, \ldots, c_{L-1})) = (m_0, \ldots, m_{L-1})$ where Bob gets the output. We only ask for one-sided security against an honest-but-curious Bob and do not require any security against a corrupt Alice – she may learn something about Bob's ciphertexts $c_0, \ldots, c_{L-1}$ during the course of the protocol.

Our protocol makes use of an SSB hash function $\mathcal{H} = (\mathsf{Gen}, H, \mathsf{Open}, \mathsf{Verify})$ with alphabet $\Sigma = \{0,1\}^{\ell_{blk}}$ where $\ell_{blk} := \ell_{ctx} + 1$. Assume the SSB hash has some corresponding output size ℓ_{hash} and opening size ℓ_{opn} (all polynomials in the security parameter λ). The protocol relies on an indistinguishability obfuscation (iO) scheme $\mathcal{O}$; see Appendix A for a definition of iO. The protocol is given in Figure 1.

Protocol 1. SFE for multi-decryption $f^{\mathcal{E},L}_{\mathsf{multi\text{-}dec}}(sk, (c_0, \ldots, c_{L-1})) = (m_0, \ldots, m_{L-1})$.

- Alice chooses $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, 0)$ and sends hk to Bob.
- Bob chooses randomness $(r_0, \ldots, r_{L-1}) \leftarrow \{0,1\}^L$. Let $x[i] = (c_i, r_i) \in \Sigma$ and $x = (x[0], \ldots, x[L-1]) \in \Sigma^L$. Bob computes $y \leftarrow H_{\mathsf{hk}}(x)$ and sends y to Alice.
- Alice creates a circuit $C = C[\mathsf{hk}, y, sk]$ as described below and obfuscates it by computing $\widetilde{C} \leftarrow \mathcal{O}(1^\lambda, C)$. She sends $\widetilde{C}$ to Bob.
- For $i = 0, \ldots, L-1$, Bob computes $\pi_i = \mathsf{Open}(\mathsf{hk}, x, i)$ and $m_i := \widetilde{C}(i, c_i, r_i, \pi)$.

Figure 1: Protocol for multi-decryption.

Given values hk (hash key), y (hash output) and sk (decryption key) define the circuit $C = C[\mathsf{hk}, y, sk]$ as follows.

$\underline{C[\mathsf{hk}, y, sk](i, c, r, \pi)}$: // Hard-coded: hash key hk, hash value y, decryption key sk.
// Input: $i \in \{0, \ldots, L-1\}$, $c \in \{0,1\}^{\ell_{ctx}}$, $r \in \{0,1\}$, $\pi \in \{0,1\}^{\ell_{opn}}$.
1. Check that $\mathsf{Verify}(\mathsf{hk}, y, i, (c, r), \pi) = \mathsf{accept}$. If not, output 0.
2. Output $\mathsf{Dec}_{sk}(c)$.

In addition, we assume that the circuit C includes some polynomial-size padding to make it sufficiently large. In particular, we also define an augmented circuit $C^{aug} = C^{aug}[\mathsf{hk}, y, sk, k, i^*]$ below, which is not used in the protocol, but is used in the proof of security. We will need to pad C so that its size matches that of C^{aug}. For the definition of C^{aug}, we assume that $f_k(x)$ is a PRF with key $k \in \{0,1\}^\lambda$, input $x \in \{0, \ldots, L-1\}$ and output $f_k(x) \in \{0,1\}$.

$\underline{C^{aug}[\mathsf{hk}, y, sk, k, i^*](i, c, r, \pi)}$: //New values: PRF key k, index $i^* \in \{0, \ldots, L-1\}$.
1. Check that $\mathsf{Verify}(\mathsf{hk}, y, i, (c, r), \pi) = \mathsf{accept}$. If not, output 0.
2. If $i \geq i^*$ output $\mathsf{Dec}_{sk}(c)$, else output $f_k(i) \oplus r$.

Theorem 3.3. *If $\mathcal{H}$ is an SSB hash and $\mathcal{O}$ is an iO scheme then Protocol 1 is a secure SFE for the functionality $f^{\mathcal{E},L}_{\mathsf{multi\text{-}dec}}$ with one-sided security against an honest-but-curious Bob. Furthermore, for any choice of encryption scheme $\mathcal{E}$, there is some polynomial $p(\lambda)$ such that for every polynomial $L(\lambda)$ the communication complexity of the above protocol for $f^{\mathcal{E},L}_{\mathsf{multi\text{-}dec}}$ is bounded by $p(\lambda)$ and is independent of $L(\lambda)$.*

Proof. Firstly, we describe the simulator S which gets Bob's inputs/outputs $\{c_i, m_i\}_{i=0}^{L-1}$ and creates a simulated view $\mathsf{sim}_B = (\{r_i\}_{i=0}^{L-1}, \mathsf{hk}, \widetilde{C})$ consisting of Bob's simulated random coins $\{r_i\}$ and the simulated protocol messages $\mathsf{hk}, \widetilde{C}$ from Alice. The simulator S chooses a random PRF key $k \leftarrow \{0,1\}^\lambda$ and sets the random coins to $r_i := f_k(i) \oplus m_i$ for $i = 0, \ldots, L-1$. It chooses the hashing key $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, L-1)$, sets $x := (x[0], \ldots, x[L-1])$ with $x[i] = (c_i, r_i)$ and computes $y := H_{\mathsf{hk}}(x)$. Finally, it creates a circuit $C' = C^{aug}[\mathsf{hk}, y, \bot, k, i^* = L]$ (containing $\bot$ in place of Alice's secret key sk) and sets $\widetilde{C} \leftarrow \mathcal{O}(1^\lambda, C')$. Note that, as a quick "sanity check", when $\pi_i = \mathsf{Open}(\mathsf{hk}, x, i)$ then $\widetilde{C}(i, c_i, r_i, \pi_i) = f_k(i) \oplus r_i = m_i$.

Let's fix some choice of Alice/Bob inputs: sk and $c_0, \ldots, c_{L-1}$, which also fixes Bob's outputs: $m_i = \mathsf{Dec}_{sk}(c_i)$.[8] Let view_B be the real-world view of an honest-but-curious Bob in the protocol when interacting with an honest Alice using the above inputs. Let $\mathsf{sim}_B \leftarrow S(\{c_i, m_i\}_{i=0}^{L-1})$ be the simulated view as described above. We need to show that the real and simulated views are indistinguishable: $\mathsf{view}_B \overset{c}{\approx} \mathsf{sim}_B$. We do so via a sequence of hybrid arguments.

Let **Hybrid** 0 be the distribution of the real-world view view_B. We define a sequence of hybrids where we make small modifications and eventually arrive at the distribution of the simulated view sim_B.

Let **Hybrid** 1 be the same as **Hybrid** 0, but instead of choosing Bob's coins r_i uniformly at random, we choose a random PRF key $k \leftarrow \{0,1\}^\lambda$ and we set $r_i = f_k(i) \oplus m_i$. It's clear that **Hybrid** 0 and **Hybrid** 1 are indistinguishable by the security of the PRF.

Let **Hybrid** 2 be the same as **Hybrid** 1, but instead of obfuscating the circuit $C = C[\mathsf{hk}, y, sk]$ on behalf of Alice, we obfuscate the augmented circuit $C_0^{aug} = C^{aug}[\mathsf{hk}, y, sk, k, i^* := 0]$ (containing Alice's key sk), by setting $\widetilde{C} \leftarrow \mathcal{O}(1^\lambda, C_0^{aug})$. Note that C and C_0^{aug} compute the same function, and also C was padded to be of the same size as C_0^{aug}. Therefore, these two hybrids are indistinguishable by the iO security of the obfuscator $\mathcal{O}$.

Let **Hybrid** 2.(i, j) for $i, j \in \{0, \ldots, L\}$ be the same as **Hybrid** 2 except that: (I) instead of obfuscating C_0^{aug}, we obfuscate $C_i^{aug} = C^{aug}[\mathsf{hk}, y, sk, k, i^* := i]$ by setting $\widetilde{C} \leftarrow \mathcal{O}(1^\lambda, C_i^{aug})$ (II) instead of choosing $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, 0)$ we choose $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, L, j)$ to be statistically binding at index j. It's easy to see that **Hybrid** 2 and 2.$(0, 0)$ are identical.

[8] These are "worst-case" inputs/outputs. We never rely on the security of the encryption scheme in our proof.

We claim that:

$$\textbf{Hybrid}\ \ 2.(i,i) \quad \overset{c}{\approx} \quad \textbf{Hybrid}\ \ 2.(i+1,i) \qquad i \in \{0,\ldots,L-1\} \tag{3}$$

$$\textbf{Hybrid}\ \ 2.(i+1,i) \quad \overset{c}{\approx} \quad \textbf{Hybrid}\ \ 2.(i+1,i+1) \qquad i \in \{0,\ldots,L-2\} \tag{4}$$

To see (3), notice that C_i^{aug} and C_{i+1}^{aug} only differ in how they evaluate inputs of the form (i, c, r, π) for which $\mathsf{Verify}(\mathsf{hk}, y, i, (c, r), \pi) = \mathsf{accept}$. By the "somewhere statistically binding" property of the SSB hash which is selected to be binding at index i, such inputs must satisfy $(c, r) = (c_i, r_i)$ where c_i is Bob's i'th input and $r_i = f_k(i) \oplus m_i$. In this case C_i^{aug} outputs $\mathsf{Dec}_{sk}(c) = m_i$ and C_{i+1}^{aug} outputs $f_k(i) \oplus r_i = m_i$, meaning that the circuits behave the same way on all inputs. Therefore, (3) follows by the iO security of the obfuscation scheme. On the other hand, (4) follows immediately from the index hiding property of the SSB hash function. Together, this shows that **Hybrid** 2 is indistinguishable from **Hybrid** $2.(L, L-1)$.

Let **Hybrid** 3 be the same as **Hybrid** $2.(L, L-1)$ but instead of obfuscating the circuit $C_L^{aug} = C^{aug}[\mathsf{hk}, y, sk, k, i^* = L]$ we obfuscate the circuit $C' = C^{aug}[\mathsf{hk}, y, \bot, k, i^* = L]$ (replacing sk with $\bot$). Since sk is never used in either circuit, their behavior is identical, and therefore the indistinguishability of these hybrids follows from the iO security of the obfuscation scheme.

Hybrid 3 is the same as the simulate view sim_B. Therefore, putting everything together, we showed that the real/simulated views are indistinguishable $\mathsf{view}_B \overset{c}{\approx} \mathsf{sim}_B$, which proves the theorem.

□

3.3 General SFE Constructions

We now leverage the protocol for multi-decryption from the previous section to get generic SFE protocols in the honest-but-curious setting. Let

$$f = \{f_\lambda : \{0,1\}^{\ell_A(\lambda)} \times \{0,1\}^{\ell_B(\lambda)} \to \{0,1\}^{L(\lambda)}\}_{\lambda \in \mathbb{N}}$$

be any efficiently computable function with ℓ_A, ℓ_B, L being some polynomials in the security parameter λ. We focus on SFE schemes where Alice has input x_A, Bob has input x_B and Bob learns the output $y = f(x_A, x_B)$.

We give two constructions. The first one achieves communication complexity $\mathsf{poly}(\lambda) + \ell_A(\lambda)$, where $\mathsf{poly}(\lambda)$ is some fixed polynomial independent of the choice of f or its parameters. In particular, the communication complexity only depends on Alice's input size ℓ_A but is independent of Bob's input-size ℓ_B or output-size L. The second construction achieves communication complexity $\mathsf{poly}(\lambda)\mathbf{CC}(f, \lambda)$ where $\mathbf{CC}(f, \lambda)$ is the communication complexity of an arbitrary insecure protocol for evaluating the function f and $\mathsf{poly}(\lambda)$ is some fixed polynomial independent of the choice of f or its parameters. Since there is always a simple insecure protocol where Alice sends her input to Bob, we have $\mathbf{CC}(f, \lambda) \leq \ell_A(\lambda)$ and in general, it may be much smaller than Alice's input size. Unfortunately, in this construction we pay with a multiplicative polynomial overhead rather than an additive one as before.

First Construction. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ be an FHE scheme with rerandomization (see Appendix B). As discussed in Appendix B, by relying on hybrid encryption we can assume without loss of generality that a ciphertext produced by $c \leftarrow \mathsf{Enc}_{pk}(x)$ is of size $|x| + \mathsf{poly}(\lambda)$, meaning that there is only an additive polynomial overhead. Our protocol is given in Figure 2.

Theorem 3.4. *Assuming that $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ is an FHE scheme with rerandomization, and that the conditions of Theorem 3.3 (Protocol 1) hold, Protocol 2 gives a*

PROTOCOL 2. General SFE Protocol for $f = \{f_\lambda : \{0,1\}^{\ell_A(\lambda)} \times \{0,1\}^{\ell_B(\lambda)} \to \{0,1\}^{L(\lambda)}\}$. Alice has input x_A, Bob has input x_B and Bob learns $y = f(x_A, x_B)$.

- Alice computes $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $c_A \leftarrow \mathsf{Enc}_{pk}(x_A)$ and sends pk, c_A to Bob.
- Bob computes $c_{out} = \mathsf{Eval}_{pk}(f(\cdot, x_B), c_A)$. Bob chooses a "one-time-pad" $k \leftarrow \{0,1\}^L$ and sets $c_{pad} = \mathsf{Eval}_{pk}(\mathsf{OTP}_k, c_{out})$ where $\mathsf{OTP}_k(y) := y \oplus k$. Finally, Bob computes $c_{frsh} \leftarrow \mathsf{Rerand}_{pk}(c_{pad})$. Let $c_{frsh} = (c_0, \ldots, c_{L-1})$ where c_i are bit-encryptions.
- Alice and Bob execute Protocol 1 for the functionality $f^{\mathcal{E},L}_{\mathsf{multi\text{-}dec}}$ where Alice has input sk and Bob has input $c_{frsh} = (c_0, \ldots, c_{L-1})$. Bob receives the output $z \in \{0,1\}^L$ and sets $y := k \oplus z$ as the output of the protocol.

Figure 2: First Construction – general protocol for $y = f(x_A, x_B)$.

secure SFE scheme for any polynomial-time computable functionality f. Furthermore, there is some fixed polynomial $p(\lambda)$ such that for every such functionality f where Alice's input size is $\ell_A(\lambda)$, the communication complexity of the protocol is given by $p(\lambda) + \ell_A(\lambda)$.

Proof. The communication complexity of the protocol follows by observation and relying on the fact that we can take any FHE scheme and ensure that the ciphertext-size only incurs an additive overhead, meaning that $|c_A| = \ell_A(\lambda) + p(\lambda)$ for some fixed polynomial $p(\lambda)$. See Appendix B for details.

To simulate an honest-but-curious Bob, the simulator chooses $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sets $c_A \leftarrow \mathsf{Enc}_{pk}(0^{\ell_A})$ to be an encryption of the all 0 string on behalf of Alice. It then runs Bob's honest protocol with fresh randomness to compute $c_{out}, c_{pad}, c_{frsh}$ as Bob would do. Finally it simulates the execution of Protocol 1 using the one-sided simulator for that protocol, where Bob has input $c_{frsh} = (c_0, \ldots, c_{L-1})$ and output $y = (y_0, \ldots, y_{L-1}) \in \{0,1\}^L$. It is clear that this is indistinguishable by the semantic security of the FHE scheme and the simulation-security of Protocol 1.

To simulate an honest-but-curious Alice, the simulator runs Alice's protocol to produce the first round messages pk, c_A. It then chooses randomness $z \leftarrow \{0,1\}^L$ and computes $c_{pad} \leftarrow \mathsf{Enc}_{pk}(z)$ and $c_{frsh} \leftarrow \mathsf{Rerand}_{pk}(c_{pad})$. Finally, it runs a real execution of Protocol 1 between Alice and Bob where it uses the input c_{frsh} on behalf of Bob. Notice that in the real protocol $z = y \oplus k$ has the same distribution as $z \leftarrow \{0,1\}^L$ in the simulation. Furthermore, in both the real protocol and the simulation, c_{frsh} is derived by running $\mathsf{Rerand}_{pk}(c_{pad})$ on some ciphertext c_{pad} such that $\mathsf{Dec}_{sk}(c_{pad}) = z$. The only difference between the real execution and the simulation is that c_{pad} is computed completely differently. By the security of rerandomization, this is indistinguishable.

□

Second Construction. Let $\pi^f_{insec} = (\pi^A, \pi^B)$ be any (insecure) protocol between Alice and Bob that evaluates the function $y = f(x_A, x_B)$ so that Bob learns y at the end of the protocol. We assume that it has some fixed round complexity $q = q(\lambda)$ and fixed communication-length in each round, independent of the particular inputs. Without loss of generality, the protocol π works as follows: Alice and Bob start out with a state that just consists of their inputs $\mathsf{state}^A_0 = x_A, \mathsf{state}^B_0 = x_B$. The protocol proceeds in q rounds where, in each round i, Alice computes $(\mathsf{msg}^A_i, \mathsf{state}^A_i) = \pi^A(\mathsf{msg}^B_{i-1}, \mathsf{state}^A_{i-1})$, sends msg^A_i to Bob, and Bob computes $(\mathsf{msg}^B_i, \mathsf{state}^B_i) =$

$\pi^B(\mathsf{msg}_i^A, \mathsf{state}_{i-1}^B)$ and sends msg_i^B to Alice (we define msg_0^B to be the empty string). At the end of the protocol, Bob's state $\mathsf{state}_q^B = f(x_A, x_B)$ contains the output of the computation.

Our SFE protocol will rely on the idea of "double encryption" using two FHE public keys pk_A, pk_B and ciphertexts of the form $c = \mathsf{Enc}_{pk_B}(\mathsf{Enc}_{pk_A}(x))$. To simplify notation, we let $\mathsf{Eval}_{pk_B,pk_A}(f, c)$ denote $\mathsf{Eval}_{pk_B}(\mathsf{Eval}_{pk_A}(f, \cdot), c)$. This corresponds to a homomorphic evaluation of the function f on the message x hidden under two layers of encryption. In particular if c is as above and $c^* = \mathsf{Eval}_{pk_B,pk_A}(f, c)$ then $\mathsf{Dec}_{pk_B}(\mathsf{Dec}_{pk_A}(c^*)) = f(x)$. The main idea of our construction is to execute the protocol π under two layers of FHE encryption with public keys pk_A, pk_B chosen by Alice and Bob respectively.[9]

PROTOCOL 3. General SFE Protocol for $f = \{f_\lambda : \{0,1\}^{\ell_A(\lambda)} \times \{0,1\}^{\ell_B(\lambda)} \to \{0,1\}^{L(\lambda)}\}$. Alice has input x_A, Bob has input x_B and Bob learns $y = f(x_A, x_B)$.

- Alice chooses $(pk_A, sk_A) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends pk_A to Bob. Bob chooses $(pk_B, sk_B) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends pk_B to Alice.
- Alice locally computes a double-encryption $c^A_{\mathsf{state},0} \leftarrow \mathsf{Enc}_{pk_B}(\mathsf{Enc}_{pk_A}(x_A))$ and Bob locally computes $c^B_{\mathsf{state},0} \leftarrow \mathsf{Enc}_{pk_B}(\mathsf{Enc}_{pk_A}(x_B))$.
- For $i = 1, \ldots, q$:
  - Alice computes $(c^A_{\mathsf{msg},i}, c^A_{\mathsf{state},i}) = \mathsf{Eval}_{pk_B,pk_A}(\pi^A, (c^B_{\mathsf{msg},i-1}, c^A_{\mathsf{state},i-1}))$ and sends $c^A_{\mathsf{msg},i}$ to Bob. (We define $c^B_{\mathsf{msg},0}$ to be the empty string.)
  - Bob computes $(c^B_{\mathsf{msg},i}, c^B_{\mathsf{state},i}) = \mathsf{Eval}_{pk_B,pk_A}(\pi^B, (c^A_{\mathsf{msg},i-1}, c^B_{\mathsf{state},i-1}))$ and sends $c^B_{\mathsf{msg},i}$ to Alice.
- Bob computes $c_{out} = \mathsf{Dec}_{sk_B}(c^B_{\mathsf{state},q})$.
  Bob chooses a 'one-time pad" key $k \leftarrow \{0,1\}^L$ and sets $c_{pad} = \mathsf{Eval}_{pk_A}(\mathsf{OTP}_k, c_{out})$ where $\mathsf{OTP}_k(y) := y \oplus k$.
  Finally, Bob computes $c_{frsh} \leftarrow \mathsf{Rerand}_{pk_A}(c_{pad})$. Let $c_{frsh} = (c_0, \ldots, c_{L-1})$ where c_i are bit-encryptions.
- Alice and Bob execute Protocol 1 for the functionality $f^{\mathcal{E},L}_{\mathsf{multi\text{-}dec}}$ where Alice has input sk_A and Bob has input $c_{fin} = (c_0, \ldots, c_{L-1})$. Bob receives the output $z \in \{0,1\}^L$ and sets $y := k \oplus z$ as the output of the protocol.

Figure 3: Second Construction– general protocol for $y = f(x_A, x_B)$.

Theorem 3.5. *Assume that $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ is an FHE scheme with rerandomization, and that the conditions of Theorem 3.3 (Protocol 1) hold. Let f be any polynomial-time functionality and let π^f be any (insecure) protocol correctly evaluating f with communication complexity $\mathbf{CC}(f, \lambda)$. Then Protocol 3 gives a secure SFE scheme for f. Furthermore, there is some fixed polynomial $p(\lambda)$ such that for every choice of f and π^f as above, the communication complexity of Protocol 3 is bounded by $p(\lambda)\mathbf{CC}(f, \lambda)$.*

Proof. The communication complexity of the protocol follows by inspection.

[9]We note that an alternate approach avoiding double-encryption and instead using distributed key-generation where Alice and Bob agree on a common FHE public key pk and get secret shares sk_A, sk_B of the corresponding secret key $sk = sk_A \oplus sk_B$ for pk is also possible.

To simulate an honest-but-curious Bob, the simulator runs the protocol between Alice and Bob honestly up until the last step (execution of Protocol 1) with one difference: it now computes Alice's initial encrypted state as $c^A_{\mathsf{state},0} \leftarrow \mathsf{Enc}_{pk_B}(\mathsf{Enc}_{pk_A}(0^{\ell}_A))$ using a dummy value 0^{ℓ_A} instead of her input x_A. Finally, it simulates the execution of Protocol 1 using the one-sided simulator for that protocol, where Bob has input $c_{frsh} = (c_0, \ldots, c_{L-1})$ and output $y = (y_0, \ldots, y_{L-1}) \in \{0,1\}^L$. It is clear that this is indistinguishable by the semantic security of the FHE scheme (with pk_A) and the simulation-security of Protocol 1.

To simulate an honest-but-curious Alice, the simulator runs the protocol between Alice and Bob honestly up until the last step (execution of Protocol 1) with one difference: it now computes Bob's initial encrypted state as $c^B_{\mathsf{state},0} \leftarrow \mathsf{Enc}_{pk_B}(\mathsf{Enc}_{pk_A}(0^{\ell}_B))$ using a dummy value 0^{ℓ_B} instead of his input x_B. In the last step, it chooses randomness $z \leftarrow \{0,1\}^L$ and computes $c_{pad} \leftarrow \mathsf{Enc}_{pk_A}(z)$ and $c_{frsh} \leftarrow \mathsf{Rerand}_{pk_A}(c_{pad})$. Finally, it runs a real execution of Protocol 1 between Alice and Bob where it uses the input c_{frsh} on behalf of Bob. Notice that in the real protocol $z = y \oplus k$ has the same distribution as $z \leftarrow \{0,1\}^L$ in the simulation. Furthermore, in both the real protocol and the simulation, c_{frsh} is derived by running $\mathsf{Rerand}_{pk_A}(c_{pad})$ on some ciphertext c_{pad} such that $\mathsf{Dec}_{sk_A}(c_{pad}) = z$. Therefore, the only difference between the real execution and the simulation is that (1) $c^B_{\mathsf{state},0}$ is computed with a dummy value and (2) c_{pad} is computed differently but encrypts the same message. The first modification is indistinguishable by the semantic security of the FHE scheme with public key pk_B. The second modification is indistinguishable by the re-randomization security of the FHE scheme with public key pk_A. □

Output for Alice. Note that our positive results also extend to the case where both Alice and Bob get the same output y or where they get different outputs y_A, y_B respectively. In particular, we can just run two sequential copies of our SFE where we reverse the roles of Alice and Bob. Using the first construction, this results in communication complexity $\mathsf{poly}(\lambda) + |x_A| + |x_B|$. Using the second construction, this results in communication complexity $\mathsf{poly}(\lambda)\mathbf{CC}(f, \lambda)$ where $\mathbf{CC}(f, \lambda)$ is now the communication complexity of the best insecure protocol evaluating f where both Alice and Bob get their correct outputs.

3.4 On The Necessity of Obfuscation

We show that some weak form of obfuscation is necessary to avoid "output-size dependence" for general SFE in the honest-but-curious setting. It's not clear if this weak form of obfuscation implies iO (seems unlikely) but nevertheless it appears highly non-trivial to achieve.

Succinct Obfuscation with CRS. Consider a notion of obfuscation intended for circuits $C : [L] \to \{0,1\}$ where the domain size L is some large polynomial. In other words, we would like to have an obfuscator $\widetilde{C} \leftarrow \mathcal{O}(1^\lambda, 1^L, C)$ that is allowed to run in time polynomial in the domain size L. This is trivial to achieve: the obfuscator just outputs $\widetilde{C} = [C(1), \ldots, C(L)]$ containing the evaluation of C on all points in its domain. Indeed, this satisfies a strong notion of black-box (BB) obfuscation where the entire obfuscated circuit $\widetilde{C} \overset{c}{\approx} S^C(1^\lambda, 1^L)$ can be simulated given black-box access to C.[10] Can we get such BB obfuscation scheme which is also *succinct*, meaning that the size of the obfuscated circuit $\widetilde{C}$ is bounded by some polynomial $\mathsf{poly}(|C|, \lambda)$ independent of L? It's easy to see that this is impossible using the same incompressibility argument as our main negative result - if the circuit C computes a PRF $f_k(i)$ for $i \in [L]$ then a succinct BB obfuscation scheme would need to compress the PRF outputs. Therefore, we add

[10] We use the term BB obfuscation to refer to this strong notion in contrast to the standard notion of "virtual black-box" (VBB) obfuscation where an attacker only outputs 1-bit predicate of the obfuscated circuit.

one more relaxation allowing both the obfuscator and the evaluator to have access to a large common random string (CRS) r, which is chosen uniformly at random. Although the size of r can depend on L, the size of the obfuscated circuit cannot.

Definition 3.6 (Succinct BB Obfuscation with CRS)**.** *A succinct BB obfuscator consists of a* PPT *obfuscator* $\widetilde{C} \leftarrow \mathcal{O}(1^\lambda, 1^L, C, r)$ *and a* PPT *evaluator* $b = \mathsf{Eval}(\widetilde{C}, i, r)$*, where* $r \leftarrow \{0,1\}^{p(\lambda,L,|C|)}$ *for some polynomial* p *is a uniformly random CRS. For correctness, we require that for all circuits* $C : [L] \to \{0,1\}$ *and all* $i \in [L]$ *we have* $\Pr[\mathsf{Eval}(\mathcal{O}(1^\lambda, 1^L, C, r), i, r) = C(i)] = 1$*. For security, we require that there exists a* PPT *simulator* S *such that for any polynomial* $L(\lambda)$ *and any circuit ensemble* $C = \{C_\lambda : [L(\lambda)] \to \{0,1\}\}$ *we have*

$$(r, \widetilde{C}) \overset{c}{\approx} S^C(1^\lambda, 1^L, |C|)$$

where $r \leftarrow \{0,1\}^{p(\lambda,L,|C|)}, \widetilde{C} \leftarrow \mathcal{O}(1^\lambda, 1^L, C, r)$*. In other words, we can simulate the obfuscated circuit if the simulator is allowed to choose the CRS* r*.*

Our positive result gives us such a succinct BB obfuscation scheme with a CRS using iO. To obfuscate C we first compute an SSB hash $z = H_{\mathsf{hk}}(r)$ of the CRS r and then obfuscate the circuit $C[z](i, r_i, \pi_i)$ which verifies that r_i is the correct value of the i'th bit of the pre-image by checking the opening π_i; if so, it outputs $C(i)$. More generally, we claim that any honest-but-curious SFE scheme with communication complexity $\mathsf{poly}(|x_A|, \lambda)$ which only depends on Alice's input size but not Bob's output size would give such obfuscation.

Theorem 3.7. *Assume the existence of an SFE in the honest-but-curious setting for general computation* $y = f(x_A, x_B)$ *where Bob gets the output* y*, and where the communication complexity from Alice to Bob is bounded by* $p(|x_A|, \lambda)$ *for some fixed polynomial* p *independent of the function* f *or its output size* L*. Then there exists a succinct BB obfuscator in the CRS model, where the size of the obfuscated circuit is* $p(|C|, \lambda)$ *independent of the domain size* L*.*

Proof. Consider an SFE protocol for the functionality $f(C, \perp) = y = (C(1), \ldots, C(L))$ where Alice gets as input some circuit C, Bob has no input, and Bob learns $C(1), \ldots, C(L)$. By assumption, we have an honest-but-curious SFE protocol with communication complexity $p(|C|, \lambda)$ from Alice to Bob.

The CRS in the obfuscation scheme will be the randomness r for Bob. To obfuscate a circuit C, the obfuscator $\mathcal{O}(1^\lambda, 1^L, C, r)$ runs an honest copy of the SFE protocol between Alice with input C and Bob with the random coins r. It sets the obfuscated circuit $\widetilde{C}$ to consist of the protocol messages from Alice to Bob. Given $\widetilde{C}$ and r, the Eval algorithm computes Bob's output $y = (C(1), \ldots, C(L))$. The security of the obfuscation scheme follows directly from that of the SFE protocol. □

It remain an interesting open problem to explore the notion of succinct BB obfuscation in the CRS further, and to see if it can be constructed under weaker assumptions than iO.

4 Lower Bounds in the Honest-But-Deterministic Setting

We now give a lower bound for communication complexity of offline/online SFE in the presence of honest-but-deterministic adversaries. In particular, we show that the online communication complexity in any such SFE protocol must exceed the Yao incompressibility entropy of the output distribution of the evaluated function f.

4.1 Yao Incompressibility Entropy

The traditional notion of Shannon entropy corresponds to how well a distribution can be compressed (on average). The notion of *Yao incompressibility entropy* [Yao82b; HLR07] extends this to the computational setting by measuring how well a distribution can be compressed when the compressor and decompressor are required to be *efficient*. Roughly speaking, the *Yao incompressibility entropy* of a distribution X is at least k if X cannot be efficiently compressed to fewer than k bits. We will rely on a version of *conditional* Yao incompressibility entropy due to Hsiao, Lu and Reyzin [HLR07]. It was shown by [HLR07] that the (conditional) Yao incompressibility entropy of a distribution X is always at least as large as its HILL pseudoentropy [HILL99], which is in turn at least as large as its min-entropy, and the gaps between these entropies can be large. Therefore, giving a lower bound in terms of Yao entropy yields the strongest results.

Definition 4.1 (Conditional Yao Incompressibility Entropy [HLR07])**.** *Let $k = k(\lambda)$ be an integer-valued function of security parameter λ. A probability ensemble $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ has* Yao incompressibility entropy at least k conditioned on $Z = \{Z_\lambda\}_{\lambda\in\mathbb{N}}$*, denoted by* $\mathrm{H}^{\mathrm{Yao}}(X|Z) \geq k$*, if for every pair of circuit-ensembles $C = \{C_\lambda\}$, $D = \{D_\lambda\}$ (called "compressor" and "decompressor") of size* $\mathsf{poly}(\lambda)$ *where C_λ has output-size at most $k(\lambda) - 1$, there exists a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr_{(x,z)\leftarrow(X_\lambda,Z_\lambda)}[D_\lambda(C_\lambda(x,z),z) = x] \leq \frac{1}{2} + \varepsilon(\lambda) \ .$$

We note that the above definition is actually somewhat weaker than the one of [HLR07]. The latter required that, if the output of the compressor has length ℓ, then the success probability of the compressor/decompressor should be at most $2^{\ell-k}+\varepsilon(\lambda)$. In our case, we only require this to hold for $\ell = k - 1$. Since considering a weaker definition makes our lower bound stronger, we will use our weaker variant which is also simpler to define and use.

Let $f : \{0,1\}^{\ell_A(\lambda)} \times \{0,1\}^{\ell_B(\lambda)} \to \{0,1\}^{L(\lambda)}$. We define the Yao incompressibility entropy of the function f as a natural extension of the concept of the above Yao incompressibility entropy for probability ensembles (Definition 4.1). In particular, it measures the incompressibility of Bob's output $Y = f(X_A, X_B)$ conditioned on Bob's input X_B, for the choice of distributions X_A, X_B which maximizes this quantity.

Definition 4.2 (Yao Incompressibility Entropy of Function)**.** *We say that a function $f : \{0,1\}^{\ell_A(\lambda)} \times \{0,1\}^{\ell_B(\lambda)} \to \{0,1\}^{L(\lambda)}$ has* Yao incompressibility entropy at least k*, denoted by $k \leq \mathrm{H}^{\mathrm{Yao}}(f)$, if there exist*

- *a probability ensemble $X_A = \{X_{A,\lambda}\}_{\lambda\in\mathbb{N}}$ of distributions over $\{0,1\}^{\ell_A(\lambda)}$ and*
- *a probability ensemble $X_B = \{X_{B,\lambda}\}_{\lambda\in\mathbb{N}}$ of distributions over $\{0,1\}^{\ell_B(\lambda)}$,*

such that the ensemble $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ defined via $Y = f(X_A, X_B)$ satisfies $k \leq \mathrm{H}^{\mathrm{Yao}}(Y|X_B)$; i.e., the Yao incompressibility entropy of Y conditioned on X_B is at least k.

4.2 Communication Complexity vs. Incompressibility Entropy

We now show a lower bound on the communication complexity of any (offline/online) SFE protocol evaluating f in the honest-but-deterministic setting in terms of the Yao incompressibility entropy of f.

Theorem 4.3. *Let $f : \{0,1\}^{\ell_A(\lambda)} \times \{0,1\}^{\ell_B(\lambda)} \to \{0,1\}^{L(\lambda)}$, and let $\Pi = (\Pi^{\text{off}}, \Pi^{\text{on}})$ be an offline/online protocol evaluating f with one-sided security against honest-but-deterministic Bob. If the Yao incompressibility entropy of f is $\mathrm{H}^{\text{Yao}}(f) \geq k$ then the communication complexity from Alice to Bob during the online phase of Π is at least k.*

Proof. Assume, by contradiction, that the Yao incompressibility entropy of f is at least k but the communication complexity from Alice to Bob during the online phase of Π is at most $k-1$. Since Π securely evaluates f in the presence of honest-but-deterministic Bob (Definition 2.2), there exists an efficient simulator $S = (S^{\text{off}}, S^{\text{on}})$ that satisfies the definition. Let X_A, X_B be distributions of Alice's and Bob's inputs that maximize $\mathrm{H}^{\text{Yao}}(Y|X_B)$ where $Y = f(X_A, X_B)$, so that $\mathrm{H}^{\text{Yao}}(Y|X_B) \geq k$. We show how to use the simulator S to efficiently compress the output distribution Y given X_B to $k-1$ bits, and successfully decompress with overwhelming probability.

Let $\rho = (\rho^{\text{off}}, \rho^{\text{on}})$ be any string of random coins used by the simulator S. On input $(y, x_B) \leftarrow (Y, X_B)$, the compressor C_ρ runs S on (y, x_B) using the randomness ρ to obtain the simulated view $(\mathsf{sim}_B^{\text{off}}, \mathsf{sim}_B^{\text{on}}) = ((m_1^{\text{off}}, \ldots, m_s^{\text{off}}), (x_B, m_1^{\text{on}}, \ldots, m_t^{\text{on}}))$ of the honest-but-deterministic adversary corrupting Bob. The compressor outputs $(m_1^{\text{on}}, \ldots, m_t^{\text{on}})$, i.e., the simulated messages from Alice to Bob during the online phase of length at most $k-1$.

On input $((m_1^{\text{on}}, \ldots, m_t^{\text{on}}), x_B)$, the decompressor D_ρ runs the simulator S^{off} with randomness ρ^{off} to create the simulated view $\mathsf{sim}_B^{\text{off}} = (m_1^{\text{off}}, \ldots, m_s^{\text{off}})$ of the honest-but-deterministic Bob in the offline phase. The decompressor outputs the implicit output y' of honest-but-deterministic Bob given the complete view $((m_1^{\text{off}}, \ldots, m_s^{\text{off}}), (x_B, m_1^{\text{on}}, \ldots, m_t^{\text{on}}))$.

Since the simulated transcript is computationally indistinguishable from the real transcript, a random compressor/decompressor pair will output a correct y with overwhelming probability, i.e., for all y, x_B:

$$\Pr_{\rho}[D_\rho(C_\rho(y, x_B), x_B) = y] \geq 1 - \mu(\lambda) \ ,$$

for a negligible μ. Hence, for all large enough λ there exists some fixed string ρ_λ and a pair of circuits $C_\lambda = C_{\rho_\lambda}, D_\lambda = D_{\rho_\lambda}$ of total size $s(\lambda) \in \mathsf{poly}(\lambda)$ with the output-size of C_λ being $k(\lambda) - 1$ such that

$$\Pr_{(y, x_B)}[D_\lambda(C_\lambda(y, x_B), x_B) = y] \geq 1 - \mu(\lambda).$$

This contradicts $\mathrm{H}^{\text{Yao}}(Y|X_B) \geq k$. □

As an immediate corollary, we get that the communication complexity during the online phase must be at least as large as the output-size for any functionality with pseudorandom output. For example, we state the following for the example of PRF evaluation discussed in Section 1.1.

Corollary 4.4. *Let $f = \{f_k : \{0,1\}^\lambda \to \{0,1\}\}_{k \in \{0,1\}^\lambda}$ be a pseudorandom function. Consider an SFE functionality for "L PRF Evaluations" where Alice has a key $k \in \{0,1\}^\lambda$, Bob has no input, and Bob gets the output $y = (f_k(1), \ldots, f_k(L))$ for some polynomial $L = L(\lambda)$. In any offline/online protocol $\Pi = (\Pi^{\text{off}}, \Pi^{\text{on}})$ for the above functionality, with one-sided security against honest-but-deterministic Bob, the online communication from Alice to Bob must be at least L bits.*

Proof. Consider the uniformly random distribution of Alice's input k. Then Bob's output $y = (f_k(1), \ldots, f_k(L))$ is pseudorandom and therefore the HILL and Yao incompressibility entropy of y is L. It follows from Theorem 4.3 that the communication complexity from Alice to Bob during the online phase of Π is at least L. □

Extension to Multi-Party SFE. Our negative results also extend to multi-party SFE. In particular, for an n-party functionality $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ where party P_i has input x_i and output y_i, we can define the i'th output entropy of f as being at least k if there exists some distribution $(X_1, \ldots, X_n)$ such that $\mathrm{H}^{\mathrm{Yao}}(Y_i|X_i) \geq k$ where $Y = f(X_1, \ldots, X_n)$. In that case, in any offline/online n-party SFE protocol that has one-sided security against a single honest-but-deterministic party P_i, the communication-complexity from all other parties to P_i must be at least k bits. This simply follows by thinking of party P_i as Bob and thinking of all of the other parties as Alice in a two-party SFE protocol.

4.3 Application: Lower Bounds for Functional Encryption

The impossibility of functional encryption with simulation based security for general circuits was first shown by Agrawal et al. [AGVW13]. This result was later extended to prove lower bounds for various related notions of functional encryption [DIJ+13; DI13; GGJS13]. In this section we show that the above lower bounds for functional encryption follow from our lower bound on communication complexity in offline/online SFE secure against honest-but-deterministic Bob.

Definition 4.5 (Functional Encryption)**.** *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a function family, where each circuit $C \in \mathcal{C}_\lambda$ takes as input a string $x \in \{0,1\}^{m(\lambda)}$ and outputs $C(x) \in \{0,1\}$. A* functional encryption scheme *$\mathcal{FE}$ for a circuit family $\mathcal{C}$ consists of four algorithms $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows:*

- *$\mathcal{FE}.\mathsf{Setup}(1^\lambda)$ is a* PPT *algorithm that takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{MPK}, \mathsf{MSK})$.*
- *$\mathcal{FE}.\mathsf{KeyGen}(\mathsf{MSK}, C)$ is a* PPT *algorithm that takes as input the master secret key MSK and a circuit $C \in \mathcal{C}_\lambda$ and outputs a corresponding secret key sk_C.*
- *$\mathcal{FE}.\mathsf{Enc}(\mathsf{MPK}, x)$ is a* PPT *algorithm that takes as input the master public key MPK and an input message $x \in \{0,1\}^{m(\lambda)}$ and outputs a ciphertext c.*
- *$\mathcal{FE}.\mathsf{Dec}(sk_C, c)$ is a deterministic algorithm that takes as input the secret key sk_C and a ciphertext c and outputs y.*

We require:

Correctness: *For all $C \in \mathcal{C}_\lambda$ and all $x \in \{0,1\}^{m(\lambda)}$,*

$$\Pr\left[y \neq C(x) \;\middle|\; \begin{array}{c}(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathcal{FE}.\mathsf{Setup}(1^\lambda), \mathsf{sk}_C \leftarrow \mathcal{FE}.\mathsf{KeyGen}(\mathsf{MSK}, C),\\ c \leftarrow \mathcal{FE}.\mathsf{Enc}(\mathsf{MPK}, x), y \leftarrow \mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_C, c)\end{array}\right] = \mathsf{negl}(\lambda)\ ,$$

where the probability is taken over the coins of $\mathcal{FE}.\mathsf{Setup}$, $\mathcal{FE}.\mathsf{KeyGen}$, and $\mathcal{FE}.\mathsf{Enc}$.

L-SIM Security: *For security, we require that there exists a simulator $S = (S_1, S_2)$ such that for every choice of the circuits $C_1, \ldots, C_L \in \mathcal{C}_\lambda$ and every $x \in \{0,1\}^{m(\lambda)}$ we have*

$$(\ \mathsf{view}_1 = (\mathsf{MPK}, \mathsf{sk}_{C_1}, \ldots, \mathsf{sk}_{C_L})\ \ ,\ \ \mathsf{view}_2 = c\ ) \ \overset{c}{\approx}\ (\ \mathsf{sim}_1\ ,\ \mathsf{sim}_2\ )$$

where $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathcal{FE}.\mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_{C_i} \leftarrow \mathcal{FE}.\mathsf{KeyGen}(\mathsf{MSK}, C_i), c \leftarrow \mathcal{FE}.\mathsf{Enc}(\mathsf{MPK}, x)$ and $(\mathsf{sim}_1, \mathsf{state}) \leftarrow S_1(1^\lambda, C_1, \ldots, C_L)$, $\mathsf{sim}_2 \leftarrow S_2(C_1(x), \ldots, C_L(x), \mathsf{state})$.

We can also consider a version where we want the scheme to satisfy L-SIM security for every polynomial $L(\lambda)$, and we call this as poly-SIM*-secure functional encryption.*

PROTOCOL 4. SFE for L PRF Evaluations: Alice holds a secret key $k \in \{0,1\}^\lambda$, Bob has no input, and Bob learns evaluations $f_k(1), \ldots, f_k(L)$.

Offline phase:

- Alice generates $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathcal{FE}.\mathsf{Setup}(1^\lambda)$ and for all $i = 1, \ldots, L(\lambda)$ Alice generates the circuit $C_i \in \mathcal{C}_\lambda$ defined as $C_i(k) = f_k(i)$, sets $\mathsf{sk}_{C_i} \leftarrow \mathcal{FE}.\mathsf{KeyGen}(\mathsf{MSK}, C_i)$ and sends $(\mathsf{MPK}, \mathsf{sk}_{C_1}, \ldots, \mathsf{sk}_{C_L})$ to Bob.

Online phase:

- Alice computes an encryption $c \leftarrow \mathcal{FE}.\mathsf{Enc}(\mathsf{MPK}, k)$ of her key $k \in \{0,1\}^\lambda$ and sends c to Bob.
- Bob outputs $y = (\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_{C_1}, c), \ldots, \mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_{C_L}, c))$.

Figure 4: Protocol for evaluating PRF f_k on L inputs.

The above notion of L-SIM security is weaker and simpler to describe than usual notions of simulation-based security for functional encryption, such as the 1-NA-SIM-secure functional encryption from Agrawal et al. [AGVW13]. This makes our results stronger and simpler. In particular, our notion corresponds to simulating an adversary $\mathcal{A}$ that makes L completely non-adaptive $\mathcal{FE}.\mathsf{KeyGen}$ queries prior to seeing the challenge ciphertext of message x. We require the simulator to simulate MPK and the secret keys sk_{C_i} before learning the outputs $C_i(x)$ on the chosen-message x. This models the fact that x is chosen by $\mathcal{A}$ later in the game (and perhaps could adaptively depend on the secret keys sk_{C_i}).

Offline/Online SFE from Functional Encryption. Let $f = \{f_k : \{0,1\}^\lambda \to \{0,1\}\}_{k \in \{0,1\}^\lambda}$ be a PRF family, and define the two-party *L PRF evaluations* functionality where Alice holds a secret key $k \in \{0,1\}^\lambda$, Bob has no input, and Bob learns evaluations $f_k(1), \ldots, f_k(L)$. We show that any L-SIM-secure functional encryption scheme $\mathcal{FE}$ for the class of circuits $\mathcal{C}_\lambda = \{C_i(k) = f_k(i) : i \in \{0,1\}^\lambda\}$ gives an offline/online two-party protocol (described in Figure 4) for this functionality. Note that the communication complexity in the online phase of Protocol 4 is equal to the ciphertext-size in $\mathcal{FE}$.

Theorem 4.6. *Let $f = \{f_k : \{0,1\}^\lambda \to \{0,1\}\}_{k \in \{0,1\}^\lambda}$ be a pseudorandom function. For any polynomial $L = L(\lambda)$, let $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be any L-SIM secure functional encryption scheme for circuit family $\mathcal{C}_\lambda = \{C_i(k) = f_k(i) : i \in \{0,1\}^\lambda\}$. Then Protocol 4 instantiated with $\mathcal{FE}$ is an offline/online protocol that evaluates the "L PRF Evaluations" functionality with security against an honest-but-deterministic (or even fully malicious) Bob.*

Proof. Since Bob does not send any messages to Alice in the protocol, there is no difference between Bob being malicious, honest-but-deterministic, or honest-but-curious in this case.

To simulate view of Bob in Protocol 4 we need to construct an offline/online simulator $S = (S^{\text{off}}, S^{\text{on}})$ by relying on the functional-encryption simulator $S' = (S'_1, S'_2)$. We simply define the offline simulator $S^{\text{off}}(1^\lambda)$ to run $S'_1(1^\lambda, C_1, \ldots, C_L)$ with the circuits $C_i(k) = f_k(i)$. We define $S^{\text{on}}(C_1(k), \ldots, C_L(k), \mathsf{state})$ to simply run $S_2(C_1(x), \ldots, C_L(x), \mathsf{state})$. The security of SFE simulation follows directly from the L-SIM security of functional encryption. □

We can now use the above theorem and our lower-bound for the communication complexity

of offline/online SFE to derive a lower bound of [AGVW13] for functional encryption.

Corollary 4.7 ([AGVW13] Corollary 1.2)**.** *There exists a circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}$ such that for every polynomial $L = L(\lambda)$, every L-SIM-secure functional encryption scheme for $\mathcal{C}$ must have ciphertext-size at least L bits. In particular there is no* poly-SIM*-secure functional encryption scheme for $\mathcal{C}$.*

Proof. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be the circuit family from Theorem 4.6. Assume, by contradiction, there exists an L-SIM-secure functional encryption scheme $\mathcal{FE}$ for $\mathcal{C}$ with ciphertext-size $L-1$. It follows from Theorem 4.6 that Protocol 4 instantiated with $\mathcal{FE}$ evaluates the "L PRF Evaluations" SFE functionality with security against honest-but-deterministic Bob, and the communication complexity from Alice to Bob during the online phase is $L-1$ bits. However, such offline/online SFE protocol contradicts Corollary 4.4 stating that the communication complexity during the online phase must be at least L bits.

The above shows that if the ciphertext size of the scheme $\mathcal{FE}$ is $L = L(\lambda)$ bits, then it already cannot be $(L+1)$-SIM secure. Therefore no functional encryption scheme can be poly-SIM-secure. □

The technique of Agrawal et al. [AGVW13] was also used by De Caro et al. [DIJ+13] in the case of fully non-adaptive adversaries (adversaries that must issue the ciphertext queries and the secret key queries simultaneously), De Caro and Iovino [DI13] for simulators with the additional power of rewinding the adversary, or Goldwasser et al. [GGJS13] for multi-input functional encryption. All of the above results can easily be derived as corollaries of our Theorem 4.3 by showing that the corresponding primitive can be used to get an SFE protocol in the offline/online model with better online communication than what we showed to be possible.

4.4 Application: Lower Bounds for Garbled Circuits

In this section we discuss the known lower bounds for garbled circuits that follow from our lower bound on communication complexity of offline/online SFE from Section 4.2.

Garbled circuits, introduced by Yao [Yao82a] in the context of secure two-party computation, allow to evaluate circuit C on input x without revealing anything about C or x besides $C(x)$. A garbling scheme takes as input circuit C and outputs $\tilde{C}$, a garbled version of C, such that $\tilde{C}(\tilde{x}) = C(x)$ for any garbled input $\tilde{x}$ corresponding to x.

The standard notion of security requires existence of an efficient simulator that given $C(x)$ outputs $\tilde{C}$ and $\tilde{x}$ indistinguishable from real pair of a garbled circuit and a garbled input. A stronger notion, usually referred to as *adaptive* security requires the simulator to simulate the garbled circuit independently of the output – i.e., the simulator gets C and outputs $\widetilde{C}$ then gets $C(x)$ and outputs $\tilde{x}$.[11] This corresponds to the fact that the input x may be chosen adaptively depending on the garbled circuit $\widetilde{C}$.

Garbled Circuits with Adaptive Security. The work of Applebaum et al. [AIKW13] gives a lower bound showing that (in general) the size of the garbled input must be at least as large as the output of the circuit if the garbling scheme provides *adaptive security*. Indeed, it is easy to see that an adaptively secure garbling scheme gives rise to an offline/online two-party SFE protocol where Alice sends the garbled circuit $\tilde{C}$ to Bob during the offline phase and the garbled input $\tilde{x}$ during the online phase, and finally Bob outputs $\tilde{C}(\tilde{x})$. Therefore, we can derive the same lower bound from our Corollary 4.4 by noting that there exists a circuit family for which

[11] Since the simulator receives the circuit C, the security notions we consider in this section provide only input privacy and not circuit privacy. This makes the negative results stronger.

the garbled input (= online communication) must be at least as large as the output-size of the garbled circuit.

Note that it is crucial for the lower bound of Applebaum et al. [AIKW13] that the security notion is adaptive. The lower bound can be circumvented if one considers the standard non-adaptive security, where the simulator receives the output of the circuit before outputting the simulated garbled circuit, and for example the Yao's construction of garbled circuits already provides garbled inputs of size independent of the output of the circuit.

Reusable Garbled Circuits. Until recently, most of the constructions of garbled circuits (including [Yao82a]) provided no security guarantee if used on multiple garbled inputs. Goldwasser et al. [GKP+13] put forward the notion of *reusable* garbled circuits and gave the first construction. In this case the standard (non-adaptive) security would require that we can simulate a garbled circuit and a series of garbled inputs $\widetilde{C}, \tilde{x}_1, \ldots, \tilde{x}_q$ given C and $C(x_1), \ldots, C(x_q)$. The work of Gentry et al. [GHRW14] shows that for reusable garbled circuits, even if we consider non-adaptive security, there exists a family of circuits for which the length of the garbled inputs must be at least as long as the length of the output of the circuit. We show how this follows as a simple corollary of our results.

Consider a circuit $C(x)$ which evaluates a PRG with seed x and output size L. A reusable garbled circuit scheme having garbled-inputs of size $L-1$ would give an *online-only* protocol for evaluating circuit $C(x_1), \ldots, C(x_q)$ on q different inputs by sending $\widetilde{C}, \tilde{x}_1, \ldots, \tilde{x}_q$. The communication complexity would be $|\widetilde{C}| + q(L-1)$. By choosing $q > |\widetilde{C}|$, this would be smaller than the total output size qL, contradicting Theorem 4.3.

5 Conclusions

We explored the communication complexity of SFE for functions with long output. We showed that the honest-but-curious setting allows for general protocols whose communication is smaller than the output size while the malicious or even honest-but-deterministic settings do not. There are several interesting open problems left to explore. One interesting problem would be to consider weaker security notions than simulation-based security. For example, perhaps we get around "output-size dependence" in the malicious setting if we allowed for an unbounded or super-polynomial simulator. We leave this question for future work.

6 Acknowledgements

The second author would like to thank Craig Gentry and Vinod Vaikuntanathan for bringing the question of output-size dependence to his attention in a conversation several years back.

References

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. "Functional Encryption: New Perspectives and Lower Bounds". In: *CRYPTO (2)*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 500–518. ISBN: 978-3-642-40083-4.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. "Differing-Inputs Obfuscation and Applications". In: *IACR Cryptology ePrint Archive* 2013 (2013), p. 689.

[AIKW13] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. "Encoding Functions with Constant Online Rate or How to Compress Garbled Circuits Keys". In: *CRYPTO (2)*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 166–184. ISBN: 978-3-642-40083-4.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. "Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE". In: *EUROCRYPT*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 483–501. ISBN: 978-3-642-29010-7.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *CRYPTO*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 1–18. ISBN: 3-540-42456-3.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. "On Extractability Obfuscation". In: *TCC*. Ed. by Yehuda Lindell. Vol. 8349. Lecture Notes in Computer Science. Springer, 2014, pp. 52–73. ISBN: 978-3-642-54241-1.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping". In: *ITCS*. Ed. by Shafi Goldwasser. ACM, 2012, pp. 309–325. ISBN: 978-1-4503-1115-1.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. "Efficient Fully Homomorphic Encryption from (Standard) LWE". In: *FOCS*. Ed. by Rafail Ostrovsky. IEEE, 2011, pp. 97–106. ISBN: 978-1-4577-1843-4.

[DI13] Angelo De Caro and Vincenzo Iovino. "On the Power of Rewinding Simulators in Functional Encryption". In: *IACR Cryptology ePrint Archive* 2013 (2013), p. 752.

[DIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. "On the Achievability of Simulation-Based Security for Functional Encryption". In: *CRYPTO (2)*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 519–535. ISBN: 978-3-642-40083-4.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. "On the Implausibility of Differing-Inputs Obfuscation and Extractable Witness Encryption with Auxiliary Input". In: *CRYPTO (1)*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Springer, 2014, pp. 518–535. ISBN: 978-3-662-44370-5.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *FOCS*. IEEE Computer Society, 2013, pp. 40–49.

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *STOC*. Ed. by Michael Mitzenmacher. ACM, 2009, pp. 169–178. ISBN: 978-1-60558-506-2.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. "Outsourcing Private RAM Computation". In: *IACR Cryptology ePrint Archive* 2014 (2014), p. 148.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". In: *CRYPTO (1)*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Springer, 2013, pp. 75–92. ISBN: 978-3-642-40040-7.

[GGJS13] Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai. "Multi-Input Functional Encryption". In: *IACR Cryptology ePrint Archive* 2013 (2013), p. 727.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. "Reusable garbled circuits and succinct functional encryption". In: *STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 555–564. ISBN: 978-1-4503-2029-0.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396.

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. "Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility". In: *EUROCRYPT*. Ed. by Moni Naor. Vol. 4515. Lecture Notes in Computer Science. Springer, 2007, pp. 169–186. ISBN: 978-3-540-72539-8.

[Kil92] Joe Kilian. "A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract)". In: *STOC*. Ed. by S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis. ACM, 1992, pp. 723–732. ISBN: 0-89791-511-9.

[NN01] Moni Naor and Kobbi Nissim. "Communication preserving protocols for secure function evaluation". In: *STOC*. Ed. by Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis. ACM, 2001, pp. 590–599. ISBN: 1-58113-349-9.

[Yao82a] Andrew Chi-Chih Yao. "Protocols for Secure Computations (Extended Abstract)". In: *FOCS*. IEEE Computer Society, 1982, pp. 160–164.

[Yao82b] Andrew Chi-Chih Yao. "Theory and Applications of Trapdoor Functions (Extended Abstract)". In: *FOCS*. IEEE Computer Society, 1982, pp. 80–91.

A Indistinguishability Obfuscation

Definition A.1 (Indistinguishability Obfuscator (iO))**.** *A uniform* PPT *machine $\mathcal{O}$ is called an* indistinguishability obfuscator *for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if the following conditions are satisfied:*

Correctness: *For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that*

$$\Pr[\widetilde{C}(x) = C(x) : \widetilde{C} \leftarrow \mathcal{O}(1^\lambda, C)] = 1 \ .$$

Security: *For any (not necessarily uniform)* PPT *distinguisher D, there exists a negligible function α such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x,then*

$$\left|\Pr[D(\mathcal{O}(1^\lambda, C_0)) = 1] - \Pr[D(\mathcal{O}(1^\lambda, C_1)) = 1]\right| \leq \alpha(\lambda) \ .$$

B Fully Homomorphic Encryption

An FHE scheme consists of PPT algorithms $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ and a fixed ciphertext size $\ell_{ctxt} = \ell_{ctxt}(\lambda)$ where:

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ generates a public/secret key.
- $c \leftarrow \mathsf{Enc}_{pk}(b)$ takes a bit $b \in \{0,1\}$ and creates a ciphertext $c \in \{0,1\}^{\ell_{ctxt}}$.
- $b \leftarrow \mathsf{Dec}_{sk}(c)$ decrypts c.
- $c^* = \mathsf{Eval}_{pk}(f, c_1, \ldots, c_n)$ is a deterministic algorithm that takes a circuit representing a function $f : \{0,1\}^n \to \{0,1\}$ and n ciphertexts $c_1, \ldots, c_n \in \{0,1\}^{\ell_{ctxt}}$ and outputs a ciphertext $c^* \in \{0,1\}^{\ell_{ctxt}}$.

We require:

Encryption Correctness: For any choice of $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, any $b \in \{0,1\}$ and any $c \leftarrow \mathsf{Enc}_{pk}(b)$ we have $\mathsf{Dec}_{sk}(c) = b$.

Evaluation Correctness: For any choice of $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, any ciphertexts $c_1, \ldots, c_n \{0,1\}^{\ell_{ctxt}}$ such that $\mathsf{Dec}_{sk}(c_i) = b_i \in \{0,1\}$, and any circuit $f : \{0,1\}^n \to \{0,1\}$, if we set $c = \mathsf{Eval}_{pk}(f, c_1, \ldots, c_n)$ then $c \in \{0,1\}^{\ell_{ctxt}}$ and $\mathsf{Dec}_{sk}(c) = f(b_1, \ldots, b_n)$.

Security: The FHE scheme is semantically secure.

We will often abuse notation and write $c \leftarrow \mathsf{Enc}_{pk}(m)$ for a longer message $m = (b_1, \ldots, b_n) \in \{0,1\}^n$ as shorthand for computing $c_i \leftarrow \mathsf{Enc}_{pk}(b_i)$ and setting $c = (c_1, \ldots, c_n) \in \{0,1\}^{n \cdot \ell_{ctxt}}$. If $f : \{0,1\}^n \to \{0,1\}^u$ is circuit with multi-bit output, we also write $c^* = \mathsf{Eval}_{pk}(f, c)$ where $c = (c_1, \ldots, c_n)$ as shorthand for computing $c_i^* = \mathsf{Eval}_{pk}(f_i, c_1, \ldots, c_n)$ where f_i computes the i'th output bit of f, and setting $c^* = (c_1^*, \ldots, c_u^*)$.

Additive Overhead Ciphertext-Size. The above definition is for a bit-encryption scheme. When encrypting a long message m, it would result in large multiplicative overhead $|c| = |m|\mathsf{poly}(\lambda)$. Fortunately, there is a generic trick which allows us to take any such FHE an reduce the encryption overhead via hybrid encryption.

Let G be a PRG with λ-bit seed and variable length output. To encrypt a long message m we select a random seed $x \leftarrow \{0,1\}^\lambda$ and use the FHE scheme to compute $c_{\text{FHE}} \leftarrow \mathsf{Enc}_{pk}(x)$ and set $c_{\text{PRG}} = G(x) \oplus m$. This results in ciphertext-size $|m| + \mathsf{poly}(\lambda)$ with an additive overhead. We can take such ciphertexts $(c_{\text{FHE}}, c_{\text{PRG}})$ and convert them into the standard FHE bit-encryptions of the message m. This is done by running $c^* = \mathsf{Eval}_{pk}(f(\cdot, c_{\text{PRG}}), c_{\text{FHE}})$ where $f(x, y) = y \oplus G(x)$. In this case $c^* = (c_1, \ldots, c_{|m|})$ where $c_i \in \{0,1\}^{\ell_{ctxt}}$ is an FHE bit-encryption of m_i. Then we can perform arbitrary homomorphic computation on the encrypted message m, by computing on the ciphertext c^*.

Note that although the initial ciphertext created by the above hybrid encryption process has additive overhead, after running a homomorphic evaluation we get a ciphertext with multiplicative overhead.

Rerandomization. Finally, we define a notion of rerandomization for FHE. This is a variant of "circuit private FHE" which is easier to work with in our case.

Definition B.1 (FHE with Rerandomization)**.** *An FHE scheme with rerandomization consists of* PPT *algorithms* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$*, where* $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$*, and* Eval *have the usual syntax. The rerandomization procedure* $c_{fresh} \leftarrow \mathsf{Rerand}_{pk}(c_{old})$ *takes a ciphertext* $c_{old} \in \{0,1\}^{\ell_{ctxt}}$ *and outputs a ciphertext* $c_{fresh} \in \{0,1\}^{\ell_{ctxt}}$ *such that* $\mathsf{Dec}_{sk}(c_{old}) = \mathsf{Dec}_{sk}(c_{fresh})$*. For security, we require that for every fixed choice of* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and every pair of ciphertexts* c, c' *such that* $\mathsf{Dec}_{sk}(c) = \mathsf{Dec}_{sk}(c')$*, we have:*

$$\mathsf{Rerand}_{pk}(c) \overset{c}{\approx} \mathsf{Rerand}_{pk}(c')$$

where the randomness is only over the coins of the Rerand *procedure. Note that* pk, sk, c, c' *are fixed in the above experiment, and we can therefore assume they are known to the distinguisher.*

All known constructions of FHE support rerandomization with statistical security.

Note. One subtlety of the above definitions is that, for simplicity, we define the correctness of homomorphic evaluation Eval and the correctness of rerandomization Rerand to work for any ciphertext c subject to $\mathsf{Dec}_{sk}(c)$ taking on some particular value. In the usual FHE constructions, the ciphertexts have some noise parameters and if the noise gets too large then correctness no longer holds. However, there is a generic method using bootstrapping to convert any ciphertext c such that $\mathsf{Dec}_{sk}(c) = m$ into a ciphertext with smaller noise. We can implicitly assume that this is done prior to each Eval and Rerand procedure to make our correctness property hold.