Laconic Function Evaluation and Applications

Willy Quach* Hoeteck Wee† Daniel Wichs‡

May 3, 2018

Abstract

We introduce a new cryptographic primitive called *laconic function evaluation* (LFE). Using LFE, Alice can compress a large circuit f into a small digest. Bob can encrypt some data x under this digest in a way that enables Alice to recover $f(x)$ without learning anything else about Bob's data. For the scheme to be laconic, we require that the size of the digest, the run-time of the encryption algorithm and the size of the ciphertext should all be small, much smaller than the circuit-size of f. We construct an LFE scheme for general circuits under the learning with errors (LWE) assumption, where the above parameters only grow polynomially with the depth but not the size of the circuit. We then use LFE to construct secure 2-party and multi-party computation (2PC, MPC) protocols with novel properties:

- We construct a 2-round 2PC protocol between Alice and Bob with respective inputs x_A, x_B in which Alice learns the output $f(x_A, x_B)$ in the second round. This is the first such protocol which is "Bob-optimized", meaning that Alice does all the work while Bob's computation and the total communication of the protocol are smaller than the size of the circuit f or even Alice's input x_A. In contrast, prior solutions based on fully homomorphic encryption are "Alice-optimized".
- We construct an MPC protocol, which allows N parties to securely evaluate a function $f(x_1, ..., x_N)$ over their respective inputs, where the total amount of computation performed by the parties *during* the protocol execution is smaller than that of evaluating the function itself! Each party has to individually *pre-process* the circuit f before the protocol starts and *post-process* the protocol transcript to recover the output after the protocol ends, and the cost of these steps is larger than the circuit size. However, this gives the first MPC where the computation performed by each party during the actual protocol execution, from the time the first protocol message is sent until the last protocol message is received, is smaller than the circuit size.

1 Introduction

We introduce a new and natural cryptographic primitive, which we call *laconic function evaluation (LFE)*. In an LFE scheme, Alice has a large circuit f, potentially containing various hard-coded data. She can deterministically compress f to derive a short digest $\mathsf{digest}_f = \mathsf{Compress}(f)$. Bob can encrypt some data x under this digest, resulting in a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{digest}_f, x)$, which Alice is able to decrypt using only her knowledge of f to recover the output $f(x) = \mathsf{Dec}(f, \mathsf{ct})$. Security ensures that Alice does not learn anything about Bob's input x beyond the output $f(x)$, as formalized via the simulation paradigm. The *laconic* aspect of LFE requires that Bob's computational complexity is small, and in particular, the size of digest_f, the run-time of the encryption algorithm $\mathsf{Enc}(\mathsf{digest}_f, x)$ and the size of the ciphertext ct should be much smaller than the circuit-size of f.

As an example, we can imagine that the FBI (acting as Alice) has a huge database D of suspected terrorists and publishes a short digest of the circuit $f_D(x)$ which checks if a person x belongs to the database D. An airline (acting as Bob) can use this digest to encrypt the identity of a passenger on their

*Northeastern University. E-mail: quach.w@husky.neu.edu

†CNRS and ENS. E-mail: wee@di.ens.fr. Supported by ERC Project aSCEND (H2020 639554).

‡Northeastern University. E-mail: wichs@ccs.neu.edu. Supported by NSF grants CNS-1314722, CNS-1413964.

flight manifest, which lets the FBI learn whether the passenger is on the suspected terrorist list, while preserving passenger privacy otherwise.[1] This is a special case of secure 2-party computation and indeed we will show that LFE has applications to general secure 2-party and multi-party computation with novel properties which weren't achievable using previously known techniques.

We emphasize that, in spite of all of the recent advances in succinct computation on encrypted data (from fully homomorphic encryption through functional encryption), the very natural notion of LFE has not been considered in the literature, nor does it follow readily from existing cryptographic tools and primitives. We discuss the most related primitives below.

Relation to Laconic OT. LFE can be seen as a generalization of the recently introduced notion of *laconic oblivious transfer (LOT)* [CDG+17]. In an LOT scheme, Alice has a large database $D \in \{0,1\}^M$ which she compresses into a short digest $\mathsf{digest}_D = \mathsf{Compress}(D)$. Bob can choose any location $i \in [M]$ and two messages m_0, m_1, and create a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{digest}_D, (i, m_0, m_1))$. Alice recovers the message $m_{D[i]}$, where $D[i]$ is the i'th bit of D, without learning anything about the other message $m_{1-D[i]}$. In other words, we can think of LOT as a highly restricted form of LFE for functions of the form $f_D^{\mathrm{LOT}}((i, m_0, m_1)) = (i, m_{D[i]})$, whereas LFE works for arbitrary circuits.[2] Using the ideas of [CDG+17], it is possible to combine LOT with garbled circuits to achieve a "half laconic" $\frac{1}{2}$LFE scheme for arbitrary circuits, where the digest is short but the run-time of the encryption algorithm and the size of the ciphertext are larger than the circuit size of f. (Essentially, Alice sends an LOT digest corresponding to a description of the circuit f and Bob sends a garbled universal circuit that takes as input f and outputs $f(x)$, along with LOT ciphertexts for the labels of the input wires.) Although such $\frac{1}{2}$LFE is already interesting and can be constructed under several different assumptions (e.g., DDH, LWE, etc.) by leveraging the recent works of [CDG+17, DG17, BLSV18], it will be insufficient for our applications which crucially rely on a fully laconic LFE scheme.

Relation to Functional Encryption. LFE also appears to be related to (succinct, single-key) *functional encryption* (FE) [SW05, BW07, KSW08, SS10, BSW11, GVW12, GVW13, GKP+13, BGG+14, GVW15, Agr17]. However, despite some similarity, the two primitives are distinct and have incomparable requirements. An FE scheme has a master authority which creates a master public key mpk and master secret key msk. For any function f, the master authority can use msk to generate a function secret key sk_f, which it can give to Alice. Bob can then use mpk to compute an encryption $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x)$ of some value x, which Alice can decrypt using sk_f to recover $f(x)$ without learning anything else about x. In a succinct FE scheme, the size of mpk, the run-time of the encryption algorithm and the ciphertext size are smaller than the circuit size of f. There are important differences between FE and LFE:

- In FE there is a master authority which is a separate party distinct from the users Alice/Bob and which gives sk_f to Alice and mpk to Bob. In LFE there is no such additional authority.
- In FE the ciphertext ct is not tied to any specific function f, but is created using a master public key mpk. Depending on which secret key sk_f Alice gets she is able to decrypt $f(x)$, but the master authority is able to learn all of x. In LFE, the ciphertext ct is created using digest_f which ties it to a specific function f, and the ciphertext does not reveal anything beyond $f(x)$ to anyone.

We do not know of any generic relationship between these two notions where either one would imply the other. Nevertheless, as we will discuss later, our concrete construction of LFE under the learning with errors assumption borrows heavily from techniques developed in the context of functional encryption.

[1] We may also desire "function privacy", discussed later, to ensure that the digest does not reveal anything about D.

[2] In fact, LOT can be seen as a restricted form of "attribute-based LFE" (AB-LFE) discussed later.

1.1 Main Results for LFE

In this work, we construct an LFE scheme under the *learning with errors (LWE)* assumption. The size of the digest, the complexity of the encryption algorithm and the size of the ciphertext only scale with the depth but not the size of the circuit. In particular, for a circuit $f : \{0,1\}^k \to \{0,1\}^\ell$ of size $|f|$ and depth d, and for security parameter λ, our LFE has the following parameters:

- The size of the digest is $\mathsf{poly}(\lambda)$ and the run-time of the encryption algorithm and the size of the ciphertext are $\widetilde{O}(k+\ell) \cdot \mathsf{poly}(\lambda, d)$.
- The run-time of the compression and the decryption algorithms is $\widetilde{O}(|f|) \cdot \mathsf{poly}(\lambda, d)$.

As in many other prior works building advanced cryptosystems under LWE, we require LWE with a sub-exponential modulus-to-noise ratio, which follows from the hardness of worst-case lattice problems with sub-exponential approximation factors and is widely believed to hold.

Necessity of a CRS. It turns out that LFE schemes require some sort of a public seed, which we refer to as a *common reference string (CRS)*. This is for the same reason that collision-resistant hash functions (CRHFs) need a public seed; in fact, it is easy to show that the compression function which maps a circuit f to a digest digest_f is a CRHF. In our case, the CRS is a *uniformly random string*, which is given as an input to all of the algorithms of the LFE. The CRS is of size $k \cdot \mathsf{poly}(\lambda, d)$. For simplicity, we will usually ignore the CRS in our simplified notation used throughout the introduction.

Selective vs. Adaptive Security. One caveat of our schemes is that we only achieve *selective* security where Bob's input x must be chosen non-adaptively before the CRS is known. We also consider *adaptive* security where Bob's input x can adaptively depend on the CRS and show that it follows from a natural and easy to state strengthening of the LWE assumption, which we call *adaptive LWE* (see the "Our Techniques" section). Currently, we can only prove the security of adaptive LWE from standard LWE via a reduction with an exponential loss of security, but it may be reasonable to assume that the actual security level of adaptive LWE is much higher than this reduction suggests.

Function Hiding. For the default notion of LFE, the compression function that computes $\mathsf{digest}_f = \mathsf{Compress}(f)$ is deterministic and the output digest_f may reveal some partial information about the circuit f. However, in many cases we may also want a *function hiding* property, meaning that the digest should completely hide the function f. In this case the compression function has to use private randomness. We show that there is a generic way to convert any LFE scheme with a deterministic compression function and without function hiding into one which has a randomized compression function and is also *statistically* function hiding. We also give an alternate more direct approach to achieving statistical function hiding for our LWE-based construction of LFE.

In a function-hiding LFE with a randomized compression function, Alice will also need to remember the random coins r that she used to create $\mathsf{digest}_f = \mathsf{Compress}(f; r)$ as a secret key, which is needed to decrypt ciphertexts created under this digest. One additional advantage of an LFE with function hiding security is that, while previously anybody (not just Alice) could decrypt a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{digest}_f, x)$ to recover $f(x)$ by just knowing the function f, the function hiding property implicitly guarantees that only Alice can recover $f(x)$ using her private randomness r, while others do not learn anything about x. This is because an external observer cannot differentiate between a correctly generated digest_f and digest_{null}, where the latter is the digest of the null function that always outputs 0. Given digest_{null} and a ciphertext ct encrypting x the attacker does not learn anything about x even given the randomness used to generate the digest.

We note that, although the compression function in function-hiding LFE is randomized, our compiler guarantees that Bob's security holds even if Alice chooses the randomness of the compression function maliciously.

1.2 Applications of LFE

As an application of LFE, we construct 2-party and multi-party computation (2PC, MPC) protocols with novel properties that weren't previously known.

Bob-Optimized 2-Round 2PC. Consider a 2-round 2PC protocol between Alice and Bob with respective inputs x_A, x_B where Alice learns the output. Without loss of generality, Alice initiates the protocol by sending the first round message to Bob and learns the output $y = f(x_A, x_B)$ after receiving the second round message from Bob.

If we didn't care about security we would have two basic *insecure* approaches to the problem. The "Alice-optimized" approach is for Alice to send her input x_A to Bob and for Bob to compute $y = f(x_A, x_B)$ and send it to Alice; Alice's computation and the total communication of the protocol are equal to $|x_A| + |y|$ while Bob's computation is equal to $|f|$. The "Bob-optimized" approach is for Alice to ask Bob to send her his input x_B and for Alice to then compute $y = f(x_A, x_B)$; Bob's computation and the total communication of the protocol are equal to $|x_B|$ while Alice's computation is equal to $|f|$. The second approach appears more natural – after all, if Alice wants to learn the output, she should do the work!

Can we get *secure* analogues of the above two approaches? Using garbled circuits and oblivious transfer, we get a protocol which is neither Alice-optimized nor Bob-optimized; the computation of both parties and the total communication are linear in the circuit size $|f|$. Using laconic OT, Alice's first message can become short but there is no improvement in either party's computation or the total communication otherwise. Using fully homomorphic encryption (FHE) [Gen09, BV11, GSW13, BV14], we can get an Alice-optimized protocol: Alice encrypts her input x_A in the first round and Bob homomorphically computes an encryption of $f(x_A, x_B)$ in the second round. Alice's computation and the total communication of the protocol are $(|x_A| + |y|) \cdot \mathsf{poly}(\lambda)$ while Bob's computation is $|f| \cdot \mathsf{poly}(\lambda)$. But can we get a Bob-optimized protocol? Previously, no such 2-round protocol was known.[3]

In this work, using LFE, we get the first "Bob-optimized" 2-round protocol where Bob's computation and the total communication are smaller than the circuit size or even the size of Alice's input – in particular, they are bounded by $(|x_B| + |y|) \cdot \mathsf{poly}(\lambda, d)$ with d being the depth of the circuit f. The linear dependence on the output size $|y|$ is inherent for any (semi-malicious) secure protocol, as shown in [HW15] but it remains an interesting open problem to get rid of the dependence on the circuit depth d. We summarize the comparison of different approaches in Figure 1.

Our protocol using LFE is extremely simple: Alice sends a digest for the function $f(x_A, \cdot)$ in the first round and Bob uses the digest to encrypt x_B in the second round. To get security for Alice, we use a function-hiding LFE scheme and we even ensure that Alice's security holds statistically. The above gives us a 2PC with semi-honest security. We can think of this as a protocol in the CRS model, or we can even allow Alice to choose the CRS on her own and send it in the first round to get a protocol in the plain model. If we think of it as a protocol in the CRS model, we even get "semi-malicious" security where corrupted parties follow the protocol but can use maliciously chosen randomness. To get fully malicious security we would need to additionally rely on succinct non-interactive zero-knowledge arguments of knowledge (ZK-SNARKs) [Mic94, Gro10, BCCT13, BCI$^+$13, GGPR13] and have Alice prove that she computed the digest correctly.

MPC with Small Online Computation. As our second application, we construct an MPC protocol that allows N parties to securely evaluate some function $f(x_1, ..., x_N)$ over their respective inputs, where the total amount of computation performed by the parties during the protocol execution is smaller than that of evaluating the function itself! Of course, the work of the computation must be performed at some point, even if we didn't care about security. In our case, the MPC protocol requires each party to

[3] If we allow additional rounds, we can use FHE to get a Bob-optimized 3-round protocol: Bob encrypts his input x_B under FHE, Alice homomorphically computes $f(x_A, x_B) \oplus k$ where k is a random one-time pad, Bob decrypts and sends the plaintext $f(x_A, x_B) \oplus k$ to Alice who recovers $y = f(x_A, x_B)$.

Approach	CRS	Communication			Computation		Assumptions
		Alice +	Bob =	Total	Alice	Bob	
Garbled Circuits	–	k_A	$\|f\|$	$\|f\|$	$\|f\|$	$\|f\|$	OT
Laconic OT	$O(1)$	$O(1)$	$\|f\|$	$\|f\|$	$\|f\|$	$\|f\|$	DDH, etc.
FHE	–	k_A	ℓ	$k_A + \ell$	$k_A + \ell$	$\|f\|$	LWE
Our Work	k_B	$O(1)$	$k_B + \ell$	$k_B + \ell$	$\|f\|$	$k_B + \ell$	LWE

Figure 1: *Summary of semi-honest 2-round 2PC where Alice holds $x_A \in \{0,1\}^{k_A}$, Bob holds $x_B \in \{0,1\}^{k_B}$, and Alice learns $y = f(x_A, x_B) \in \{0,1\}^{\ell}$. We suppress multiplicative factors that are polynomial in the security parameter λ or, for the last row, the circuit depth d.*

individually *pre-process* the function f step before the protocol starts and to *post-process* the protocol transcript and recover the output after the protocol ends, where the computational complexity of these steps can exceed the circuit size of f. The pre-processing step is deterministic and can be performed once per function being evaluated and reused across many executions. However, the main novelty of our MPC is that the total computation performed by the parties *online*, from the time they send the first protocol message and until they receive the last protocol message, is much smaller than the circuit size of f. This also implies a correspondingly small communication complexity of the protocol.

(We note there are prior MPC protocols that have a separate "offline" phase, which occurs before the inputs of the computation are known, and an "online" phase which occurs after the inputs are known and where the online phase is very efficient. However, in these schemes the term "offline" is a misnomer since the offline phase involves running a protocol and interacting with the other parties. For example, the parties may run an MPC protocol to compute a garbled circuit for the function f in the offline phase and then, once their inputs are known, they run another much more efficient MPC protocol to compute the garbled input for the garbled circuit. In contrast, ours is the first MPC that has a truly offline pre-processing and post-processing phase, where the parties do not interact with each other, while the computational complexity of the entire online phase that involves interaction is smaller than the circuit size.)

Our MPC construction uses an LFE scheme and proceeds as follows:

Pre-processing (offline). Each party individually pre-processes the circuit f by locally computing $\mathsf{digest}_f = \mathsf{Compress}(f)$. We rely on an LFE scheme with deterministic compression so all parties compute the same digest. This step can be performed once and can be reused across many executions.

Actual protocol (online). The actual protocol execution between the N parties holding inputs x_i just invokes a generic MPC protocol to compute an LFE ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{digest}_f, (x_1, \ldots, x_N))$ which encrypts their joint inputs. Here, we use the fact that the run-time of Enc is small.

Post-processing (offline). After the protocol finishes, each party individually does a post-processing step in which it runs the LFE decryption algorithm on ct to recover the output $f(x_1, \ldots, x_N)$.

The computational complexity of the pre-processing and post-processing steps is $|f| \cdot \mathsf{poly}(\lambda, d)$, where d is the depth of the circuit f. The computational complexity of each party in the online protocol execution and the total communication complexity are only $\mathsf{poly}(\lambda, d, k, \ell, N)$, where k is the input size of each party and ℓ is the output size and N is the number of parties. In particular, the communication and computational complexity of the online phase can be much smaller than the circuit size of f. We inherit semi-honest or fully-malicious security depending on the MPC used in online phase. We think of the above protocol as being in the CRS model. However, in the case of semi-honest security, we can also think of it as a protocol in the plain model by having a single designated party (e.g., party 1) choose the CRS for the LFE and compute the digest digest_f (no other parties do any pre-processing) and use these as its inputs to the MPC protocol executed in the online phase.

Improved 2-Round MPC. By taking our scheme from the previous paragraph and using a 2-round MPC [MW16, PS16, BP16, GS17, BL18, GS18] in the online phase (without any additional efficiency constraints), we get a 2-round MPC where the total computation performed by each party (including pre-processing and post-processing) is $|f| \cdot \mathsf{poly}(\lambda, d) + \mathsf{poly}(\lambda, k, \ell, d, N)$ and the total communication is $\mathsf{poly}(\lambda, k, \ell, d, N)$. In the case of semi-honest security, if we use a 2-round MPC in the plain model [GS17, BL18, GS18] in the online phase we get a 2-round MPC in the plain-model with the above efficiency. In all prior 2-round MPC protocols with N parties (even with semi-honest security, even in the CRS model), the computation performed by each party is at least $|f| \cdot N^2$. Therefore our result gives improved computational efficiency for 2-round MPC even if we did not distinguish between online and offline work.

1.3 Our Techniques

Our construction of LFE relies on an adaptation of techniques developed in the context of attribute-based and functional encryption [BGG$^+$14, GKP$^+$13]. The construction proceeds in two steps. We start by considering LFE for a restricted class of functionalities, which we call *attribute-based LFE (AB-LFE)* in analogy to attribute-based encryption (ABE). In an AB-LFE, Alice computes a digest digest_f for a function f. Bob computes a ciphertext $\mathsf{Enc}(\mathsf{digest}_f, x, \mu)$ with an attribute x and a message μ such that Alice recovers μ if $f(x) = 0$ and otherwise doesn't learn anything about μ. (For technical reasons, it will be easier to use the semantics where 0 denotes "qualified to decrypt" and 1 denotes "unqualified to decrypt".) However, the attribute x is always revealed to Alice. We can think of AB-LFE as an LFE for the "conditional disclosure functionality" $\mathsf{CDF}[f](x, \mu)$, which outputs (x, μ) if $f(x) = 0$ and $(x, \perp)$ otherwise. As our first step, we construct AB-LFE under the LWE assumption. As our second step, we show how to generically compile any AB-LFE into an LFE by additionally relying on fully homomorphic encryption.

First Step: Constructing AB-LFE. Our construction adapts the techniques developed by Boneh et al. [BGG$^+$14] to construct attribute-based encryption (ABE). In some sense, our construction of AB-LFE is simpler than that of ABE and we essentially avoid relying on "lattice trapdoors". Our proof of security also departs significantly from that of [BGG$^+$14] and appears to be simpler since we avoid embedding lattice trapdoors in the CRS. One advantage of our simplified proof is that, while for both ABE and AB-LFE we only get *selective* security under LWE, for AB-LFE (but not ABE) our proof extends to showing that *adaptive* security follows from a simple to state and natural "adaptive LWE" assumption, which we describe below. We note that, although we manage to construct AB-LFE using similar techniques as previously used in constructing of ABE, there does not appear to be a "black-box" relationship between these primitives where one would imply the other.

We rely on two algorithms $\mathsf{EvalPK}, \mathsf{EvalCT}$ that were defined by [BGG$^+$14]. Let $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ be a fixed "gadget matrix" from [MP12]. Let $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ be arbitrary matrices, $\mathbf{b}_i \in \mathbb{Z}_q^m$ be vectors, $f : \{0,1\}^k \to \{0,1\}$ be some circuit and $x \in \{0,1\}^k$ be an input.

- $\mathsf{EvalPK}(f, \{\mathbf{A}_i\}_{i \in [k]})$ outputs a matrix $\mathbf{A}_f \in \mathbb{Z}_q^{n \times m}$.
- $\mathsf{EvalCT}(f, \{\mathbf{A}_i\}_{i \in [k]}, \{\mathbf{b}_i\}_{i \in [k]}, x)$ outputs a vector $\mathbf{b}_f \in \mathbb{Z}_q^n$.

These algorithms are deterministic and have the property that:

$$\text{if} \quad \{\mathbf{b}_i = \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G}) + \mathbf{e}_i\}_{i \in [k]} \quad \text{then} \quad \mathbf{b}_f = \mathbf{s}^\top(\mathbf{A}_f - f(x)\mathbf{G}) + \mathbf{e}^* \tag{1}$$

where $\mathbf{e}_i$ and $\mathbf{e}^*$ are some "small" errors. Our basic AB-LFE scheme works as follows.

- The CRS consists of uniformly random matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$ with $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$.

- To compress a circuit $f : \{0,1\}^k \to \{0,1\}$ we set the digest to be $\mathsf{digest}_f = \mathbf{A}_f = \mathsf{EvalPK}(f, \{\mathbf{A}_i\}_{i\in[k]})$.
- The encryption algorithm $\mathsf{Enc}(\mathsf{digest}_f, x, \mu)$ encrypts a message $\mu \in \{0,1\}$ with respect to an attribute $x \in \{0,1\}^k$ under a digest $\mathsf{digest}_f = \mathbf{A}_f$. It chooses a random LWE secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and computes LWE samples $\mathbf{b}_i = \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G}) + \mathbf{e}_i$. It also chooses a random "short" vector $\mathbf{t}$ and sets $\mathbf{d} = \mathbf{A}_f \cdot \mathbf{t}$. Lastly it sets $\beta = \langle \mathbf{s}, \mathbf{d}\rangle + e' + \mu \cdot \lfloor q/2 \rceil$ and outputs the ciphertext $\mathsf{ct} = (\{\mathbf{b}_i\}, \beta, \mathbf{t}, x)$.
- The decryption algorithm computes $\mathbf{b}_f = \mathsf{EvalCT}(f, \{\mathbf{A}_i\}_{i\in[k]}, \{\mathbf{b}_i\}_{i\in[k]}, x)$. By (1), we have $\mathbf{b}_f = \mathbf{s}^\top(\mathbf{A}_f - f(x)\mathbf{G}) + \mathbf{e}^*$. If $f(x) = 0$ then this allows us to recover the message by computing $\beta - \langle \mathbf{b}_f, \mathbf{t}\rangle \approx \mu \cdot \lfloor q/2 \rceil$.

To prove security, we assume that $f(x) = 1$ and need to show that the ciphertext doesn't reveal anything about the encrypted message μ. We can rely on the LWE assumption with a uniformly random secret $\mathbf{s}$ and coefficient $(\mathbf{A}_i - x_i\mathbf{G})$ to argue that the samples $\mathbf{b}_i$ are indistinguishable from uniform. However, to argue that β is also indistinguishable from uniform is slightly more complex. Firstly, we note that by (1), we have $\beta \approx \langle \mathbf{b}_f, \mathbf{t}\rangle + \mathbf{s}^\top\mathbf{G}\mathbf{t} + e' + \mu \cdot \lfloor q/2 \rceil$. Secondly, if we set $\mathbf{u} = \mathbf{G}\mathbf{t}$, then $\mathbf{u}$ is uniformly random and $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\mathbf{u})$ can be efficiently sampled from $\mathbf{u}$, so we can efficiently sample β given $\mathbf{u}$, $\langle \mathbf{s}, \mathbf{u}\rangle + e' + \mu \cdot \lfloor q/2 \rceil$. Therefore, when we use the LWE assumption, we also get one additional sample with the coefficients $\mathbf{u}$ which we use to argue that β is indistinguishable from uniform. There is some subtlety in ensuring that the noise distribution in β is correct and we solve this using the standard "noise smudging" technique.

Adaptively Secure AB-LFE from Adaptive LWE. The above proof shows selective security, where the adversary chooses the attribute x ahead of time before seeing the CRS, but breaks down in the case of adaptive security. The issue is that, if the attribute x is chosen adaptively after the adversary sees the CRS $= \{\mathbf{A}_i\}$, then we can no longer argue that the LWE coefficients $(\mathbf{A}_i - x_i\mathbf{G})$ are uniformly random. However, the adversary has extremely limited ability to manipulate the coefficients. We formulate a new but natural "adaptive LWE" assumption where the adversary is first given matrices $\{\mathbf{A}_i\}_{i\in[k]}$, then adaptively chooses a value $x \in \{0,1\}^k$, and has to distinguish between LWE samples $\mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G}) + \mathbf{e}_i$ and uniformly random values. Our proof above shows adaptive security of AB-LFE under adaptive LWE.

What can we say about adaptive LWE? A simple guessing argument allows us to prove the security of adaptive LWE based on standard LWE by incurring a 2^k security loss in the reduction. In other words, adaptive LWE follows from the sub-exponential security of standard LWE, if we choose the parameters appropriately. However, it seems plausible to assume that adaptive LWE has a much higher level of security than this reduction suggests. As far as we can tell, it seems reasonable to assume that it could have essentially the same level of security as standard LWE and perhaps there is a reduction which only incurs a polynomial loss. We leave this fascinating question for future work.

Another interesting question is whether the adaptive LWE assumption could be useful in proving adaptive security in other contexts, most notably for attribute-based encryption (ABE) of Boneh et al. [BGG+14]. As far as we can tell, this does not appear to be the case, and the reason that we are able to prove adaptive security of AB-LFE under adaptive LWE is that our proof differs significantly from that of the Boneh et al. ABE and does not rely on embedding lattice trapdoors in the CRS. Nevertheless, we leave this as an interesting possibility for future work.

Second Step: From AB-LFE to LFE. As our second step, we show how to generically compile AB-LFE to LFE using fully-homomorphic encryption (FHE). This compiler is essentially identical to that of Goldwassser et al. [GKP+13], which showed how to compile attribute-based encryption (ABE) to (single key) functional encryption (FE). In addition to FHE, the compiler relies on garbled circuits which can be constructed from one-way functions.

The high level idea is that, in the LFE scheme, the encryptor Bob first uses an FHE to encrypt his input x, resulting in an FHE ciphertext $\widehat{x}$. He then constructs a garbled circuit $\widehat{C}$ which has the

FHE secret key hard-coded inside of it and performs an FHE decryption. Lastly, he uses an AB-LFE scheme to encrypt all of the labels of the input wires of the garbled circuit with respect to the attribute $\widehat{x}$ in such a way that Alice recovers exactly the garbled labels that correspond to the homomorphically evaluated ciphertext $\widehat{f(x)} = \mathsf{FHE.Eval}(f, \widehat{x})$. Bob sends $\widehat{x}, \widehat{C}$ and the AB-LFE ciphertexts to Alice as his LFE ciphertext.

Optimizing for Long Output and Final Parameters. The above ideas, combined together, give an LFE scheme for circuits $f : \{0,1\}^k \to \{0,1\}$ with 1-bit output and depth d, where the digest is of size $\mathsf{poly}(\lambda, d)$ and the encryption run-time and ciphertext size is $k \cdot \mathsf{poly}(\lambda, d)$. The compression and decryption run-time is $|f| \cdot \mathsf{poly}(\lambda, d)$.

A naive way to get an LFE for circuits $f : \{0,1\}^k \to \{0,1\}^\ell$ with ℓ-bit output is to invoke a separate LFE for each output bit. This would blow up all of the efficiency measures by a multiplicative factor of ℓ. It turns out that we can do better. We do so by first constructing a "multi-bit output AB-LFE" that allows Alice to compute a digest digest_f for a circuit $f : \{0,1\}^k \to \{0,1\}^\ell$ with ℓ-bit output and for Bob to create a single ciphertext with an attribute x and messages $\mu_1, \ldots, \mu_\ell$ such that Alice recovers μ_i if and only if the i'th output bit of $f(x)$ is 0. We show how to modify our LWE based AB-LFE construction to get a "multi-bit output AB-LFE" where the encryption run-time and the ciphertext size is $(k+\ell) \cdot \mathsf{poly}(\lambda, d)$ and the compression/decryption run-times remains $|f| \cdot \mathsf{poly}(\lambda, d)$. However, the digest size in this construction grows by a factor of ℓ to $\ell \cdot \mathsf{poly}(\lambda, d)$. We then show how to compress the digest further to just $\mathsf{poly}(\lambda)$ independent of ℓ, d. Essentially, instead of using the original AB-LFE digest as is, we employ an additional layer of laconic OT (LOT) and give out an LOT digest of the AB-LFE digest. The encryptor creates a garbled version of the underlying AB-LFE encryption algorithm and encrypts the labels for the garbled circuit under LOT.

Combining all of the above we get an LFE scheme where the size of the CRS is $k \cdot \mathsf{poly}(\lambda, d)$, the size of the digest is $\mathsf{poly}(\lambda)$, the run-time of the encryption algorithm and the size of the ciphertext are $\widetilde{O}(k+\ell) \cdot \mathsf{poly}(\lambda, d)$ and the run-time of the compression and the decryption algorithms is $\widetilde{O}(|f|) \cdot \mathsf{poly}(\lambda, d)$.

Additional Results. In Section 5 we describe how to generically add function hiding security to LFE and in Appendix A we give a more direct construction of function-hiding LFE under LWE. In Appendix B, we show that a CRS is necessary for LFE. In Appendix C, we present LFE protocols for the class of linear functions under the DDH assumption (and its generalization to the k-Linear assumption) in prime-order cyclic groups. In Appendix D, we present a more direct construction of LFE protocols for circuits under LWE using the "dual use" techniques in [BTVW17].

2 Preliminaries

2.1 Notations

We will denote by λ the security parameter. The notation $\mathsf{negl}(\lambda)$ denotes any function f such that $f(\lambda) = \lambda^{-\omega(1)}$, and $\mathsf{poly}(\lambda)$ denotes any function f such that $f(\lambda) = \mathcal{O}(\lambda^c)$ for some $c > 0$. For a probabilistic algorithm $\mathsf{alg}(\mathsf{inputs})$, we might explicit the randomness it uses by writting $\mathsf{alg}(\mathsf{inputs}; \mathsf{coins})$. We will denote vectors by bold lower case letters (e.g. $\mathbf{a}$) and matrices by bold upper cases letters (e.g. $\mathbf{A}$). We will denote by $\mathbf{a}^\top$ and $\mathbf{A}^\top$ the transposes of $\mathbf{a}$ and $\mathbf{A}$, respectively. We will denote by $\lfloor x \rceil$ the nearest integer to x, rounding towards 0 for half-integers. If $\mathbf{x}$ is a vector, $\lfloor \mathbf{x} \rceil$ will denote the rounded value applied component-wise.

We define the statistical distance between two random variables X and Y over some domain Ω as: $\mathbf{SD}(X, Y) = \frac{1}{2} \sum_{w \in \Omega} |X(w) - Y(w)|$. We say that two ensembles of random variables $X = \{X_\lambda\}, Y = \{Y_\lambda\}$ are *statistically indistinguishable*, denoted $X \overset{s}{\approx} Y$, if $\mathbf{SD}(X_\lambda, Y_\lambda) \leq \mathsf{negl}(\lambda)$.

We say that two ensembles of random variables $X = \{X_\lambda\}$, and $Y = \{Y_\lambda\}$ are *computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if, for all (non-uniform) PPT distinguishers Adv, we have $|\Pr[\mathsf{Adv}(X_\lambda) = 1] - \Pr[\mathsf{Adv}(Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

2.2 Learning With Errors

Definition 2.1 (B-bounded distribution)**.** *We say that a distribution χ over $\mathbb{Z}$ is B-bounded if*

$$\Pr[\,\chi \in [-B, B]\,] = 1.$$

We recall the definition of the (decision) *Learning with Errors* problem, introduced by Regev ([Reg05]).

Definition 2.2 ((Decision) Learning with Errors ([Reg05]))**.** *Let $n = n(\lambda)$ and $q = q(\lambda)$ be integer parameters and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The Learning with Errors (LWE) assumption $LWE_{n,q,\chi}$ states that for all polynomials $m = \mathsf{poly}(\lambda)$ the following distributions are computationally indistinguishable:*

$$(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}) \overset{c}{\approx} (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m, \mathbf{u} \leftarrow \mathbb{Z}_q^m$.

Just like many prior works, we rely on LWE security with the following range of parameters. We assume that for any polynomial $p = p(\lambda) = \mathsf{poly}(\lambda)$ there exists some polynomial $n = n(\lambda) = \mathsf{poly}(\lambda)$, some $q = q(\lambda) = 2^{\mathsf{poly}(\lambda)}$ and some $B = B(\lambda)$-bounded distribution $\chi = \chi(\lambda)$ such that $q/B \geq 2^p$ and the $LWE_{n,q,\chi}$ assumption holds. Throughout the paper, the *LWE assumption* without further specification refers to the above parameters. The *sub-exponentially secure LWE* assumption further assumes that $LWE_{n,q,\chi}$ with the above parameters is sub-exponentially secure, meaning that there exists some $\varepsilon > 0$ such that the distinguishing advantage of any polynomial-time distinguisher is $2^{-\lambda^\varepsilon}$.

The works of [Reg05, Pei09] showed that the (sub-exponentially secure) LWE assumption with the above parameters follows from the worst-case (sub-exponential) quantum hardness SIVP and classical hardness of GapSVP with sub-exponential approximation factors.

2.3 Lattice tools

Noise smudging. We will use the following fact.

Lemma 2.3 (Smudging Lemma (e.g., [AJL+12]))**.** *Let $B = B(\lambda), B' = B'(\lambda) \in \mathbb{Z}$ be parameters and and let $e_1 \in [-B, B]$ be an arbitrary value. Let $e_2 \leftarrow [B', B']$ be chosen uniformly at random. Then the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$ as long as $B/B' = \mathsf{negl}(\lambda)$.*

Gadget Matrix [MP12]. For an integer $q \geq 2$, define: $\mathbf{g} = (1, 2, \cdot, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{1 \times \lceil \log q \rceil}$. The *Gadget Matrix* $\mathbf{G}$ is defined as $\mathbf{G} = \mathbf{g} \otimes \mathbf{I_n} \in \mathbb{Z}_q^{n \times m}$ where $n \in \mathbb{N}$ and $m = n\lceil \log q \rceil$. There exists an efficiently computable deterministic function $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \{0,1\}^m$ such for all $\mathbf{u} \in \mathbb{Z}_q^n$ we have $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{u}) = \mathbf{u}$. We let $\mathbf{G}^{-1}(\$)$ denote the distribution obtained by sampling $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ uniformly at random and outputting $\mathbf{t} = \mathbf{G}^{-1}(\mathbf{u})$.

Lattice Evolution. We rely on the following algorithms introduced in [BGG+14].

Claim 2.4 ([BGG+14])**.** *Let $n \in \mathbb{N}$ and $m = n\lceil \log q \rceil$. There exists two deterministic algorithms* EvalPK *and* EvalCT *with the following syntax:*

- $\mathsf{EvalPK}(C, \mathbf{A}_1, \ldots, \mathbf{A}_k)$ *takes as input a circuit $C : \{0,1\}^k \to \{0,1\}$ and matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$, and outputs a matrix $\mathbf{A}_C \in \mathbb{Z}_q^{n \times m}$;*

- $\mathsf{EvalCT}(C, \mathbf{A}_1, \ldots, \mathbf{A}_k, \mathbf{b}_1, \ldots \mathbf{b}_k, x)$ *takes as input a circuit* $C : \{0,1\}^k \to \{0,1\}$*, matrices* $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$*, vectors* $\mathbf{b}_i \in \mathbb{Z}_q^m$ *and an input* $x \in \{0,1\}^k$*, and outputs a vector* $\mathbf{b}_C \in \mathbb{Z}_q^m$*;*

such that if there exists some $\mathbf{s} \in \mathbb{Z}_q^n$ *such that:*

$$\forall i \leq k, \quad \mathbf{b}_i = \mathbf{s}^\top (\mathbf{A}_i - x_i \mathbf{G}) + \mathbf{e}_i \text{ with } \|\mathbf{e}_i\|_\infty \leq B,$$

then

$$\mathbf{b}_C = \mathbf{s}^\top (\mathbf{A}_C - C(x)\mathbf{G}) + \mathbf{e}_C \text{ where } \|\mathbf{e}_C\|_\infty \leq (m+1)^d \cdot B.$$

Furthermore, the run-time of $\mathsf{EvalPK}, \mathsf{EvalCT}$ *is* $|C| \cdot \mathsf{poly}(n, \log q)$*.*

We can extend the above algorithms to support circuits $C : \{0,1\}^k \to \{0,1\}^\ell$ with multi-bit output. In this case, we have that:

- $\mathsf{EvalPK}(C, \mathbf{A}_1, \ldots, \mathbf{A}_k)$ outputs ℓ matrices $\{\mathbf{A}_{C_j}\}_{j \leq \ell} \in (\mathbb{Z}_q^{n \times m})^\ell$;
- $\mathsf{EvalCT}(C, \mathbf{A}_1, \ldots, \mathbf{A}_k, \mathbf{b}_1, \ldots \mathbf{b}_k, x)$ outputs ℓ vectors $\{\mathbf{b}_{C_j}\}_{j \leq \ell} \in (\mathbb{Z}_q^m)^\ell$.

The output is identical to having run $\mathsf{EvalPK}, \mathsf{EvalCT}$ separately for each output bit of C. However, by processing the entire circuit in one shot (rather than looking at each output bit separately), we ensure that the run-time of $\mathsf{EvalPK}, \mathsf{EvalCT}$ remains $|C| \cdot \mathsf{poly}(n, \log q)$ instead of $|C| \cdot \ell \cdot \mathsf{poly}(n, \log q)$.

2.4 Fully Homomorphic Encryption

A leveled Fully Homomorphic Encryption scheme (FHE) is a set of algorithms ($\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}$, $\mathsf{FHE.Dec}, \mathsf{FHE.Eval}$) satisfying the following properties:

- **Security:** ($\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}$) is a semantically secure encryption scheme;
- **Perfect correctness:** For all λ, all $C : \{0,1\}^k \to \{0,1\}^\ell$ of depth d and $x \in \{0,1\}^k$:

$$\Pr\left[\mathsf{FHE.Dec}_{\mathsf{hsk}}\left(\mathsf{FHE.Eval}(C, \mathsf{FHE.Enc}_{\mathsf{hpk}}(x))\right) = C(x) \quad | \quad (\mathsf{hpk}, \mathsf{hsk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)\right] = 1;$$

- **Compactness:** If $C : \{0,1\}^k \to \{0,1\}^\ell$ is a circuit of depth d, then the output length of $\mathsf{FHE.Eval}(C, \cdot)$ should be $\ell \cdot \mathsf{poly}(\lambda, d)$.
- **Efficiency:** If $C : \{0,1\}^k \to \{0,1\}^\ell$ is a circuit of depth d, then $\mathsf{FHE.Eval}(C, \cdot)$, which takes as input a ciphertext ct, and outputs $\mathsf{FHE.Eval}(C, \mathsf{ct})$ can be computed by a circuit of depth $d \cdot \mathsf{polylog}(\lambda)$ and size $|C| \cdot \mathsf{poly}(\lambda, d)$.

Such FHE schemes are known to exist under the LWE assumption (e.g., [BV11, GSW13]).

2.5 Garbled Circuits

We define here garbled circuits, originally introduced by Yao ([Yao82]), and there are now many variants in the literature ([BHR12]). The following formalization is heavily inspired by the one used in [GKP+13].

A Garbling Scheme is a set of algorithms ($\mathsf{GC.Garble}, \mathsf{GC.Eval}$) such that:

- $\mathsf{GC.Garble}(1^\lambda, C)$ takes as input the security parameter λ, a circuit $C : \{0,1\}^k \to \{0,1\}^\ell$, and outputs a garbled circuit Γ and a set of labels $\{L_i^0, L_i^1\}_{i \leq k}$.
- $\mathsf{GC.Eval}(\Gamma, \{L_i\}_{i \leq k})$ takes as input a garbled circuit and a subset of labels, and outputs a value $y \in \{0,1\}^\ell$.
- Algorithms $\mathsf{GC.Garble}$ and $\mathsf{GC.Eval}$ satisfy the following properties:

Correctness. We have for all circuits $C : \{0,1\}^k \to \{0,1\}^\ell$ and for all $x \in \{0,1\}^k$:

$$\Pr[\, C(x) = y \quad | \quad (\Gamma, \{L_i^0, L_i^1\}_{i\le k}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C), y \leftarrow \mathsf{GC.Eval}(\Gamma, \{L_i^{x_i}\}_{i\le k})\,] = 1,$$

where x_i denotes the ith bit of x.

Circuit and Input privacy. Define the two following experiments:

$\text{EXP}_{GC}^{Real}(1^\lambda)$:

1. $(x, C) \leftarrow \mathsf{Adv}(1^\lambda)$
2. $(\Gamma, \{L_i^0, L_i^1\}_{i\le k}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$
3. $b \in \{0,1\} \leftarrow \mathsf{Adv}(\Gamma, \{L_i^{x_i}\}_{i\le k})$
4. Output b.

$\text{EXP}_{GC}^{Ideal}(1^\lambda)$:

1. $(x, C) \leftarrow \mathsf{Adv}(1^\lambda)$
2. $(\tilde{\Gamma}, \{\tilde{L}_i\}_{i\le k}) \leftarrow \mathsf{Sim}_{GC}(1^\lambda, |C|, |x|, C(x))$
3. $b \in \{0,1\} \leftarrow \mathsf{Adv}(\tilde{\Gamma}, \{\tilde{L}_i\}_{i\le k})$
4. Output b.

We say that $(\mathsf{GC.Garble}, \mathsf{GC.Eval})$ is circuit and input private if there exists a PPT simulator Sim_{GC} such that for all stateful PPT adversary Adv, we have:

$$\left| \Pr\left[\text{EXP}_{GC}^{Real}(1^\lambda) = 1 \right] - \Pr\left[\text{EXP}_{GC}^{Ideal}(1^\lambda) \right] \right| \le \mathsf{negl}(\lambda).$$

Efficiency. For any circuit C, and $(\Gamma, \{L_i^0, L_i^1\}_{i\le k}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$, we have the following properties:

- $\mathsf{GC.Garble}(1^\lambda, C)$ has complexity $|C| \cdot \mathsf{poly}(\lambda)$;
- $\mathsf{GC.Eval}(\Gamma, \cdot)$ has complexity $|C| \cdot \mathsf{poly}(\lambda)$;
- Γ is of size $|C| \cdot \mathsf{poly}(\lambda)$;
- L_i^b is of size $\mathsf{poly}(\lambda)$ for all $i \le k$ and $b \in \{0,1\}$.

2.6 Entropy and Extractors

The *min-entropy* of a random variable X, is defined as $\mathbf{H}_\infty(X) = -\log(\max_x \Pr[X = x])$. The *(average) conditional min-entropy* [DORS08] of a random variable X conditioned on Y, is defined as $\mathbf{H}_\infty(X|Y) = -\log(\mathbb{E}_y \max_x \Pr[X = x|Y = y])$.

Lemma 2.5 ([DORS08])**.** *If X, Y, Z are jointly distributed random variables and the support of Y is $\mathcal{Y}$ then $\mathbf{H}_\infty(X|Y, Z) \ge \mathbf{H}_\infty(X|Z) - \log(|\mathcal{Y}|)$.*

We say that $\mathsf{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ is a (k, ϵ)-*extractor*, if for all joint distribution (X, Z) where X is supported over $\mathcal{X}$ and $\mathbf{H}_\infty(X|Z) \ge k$, we have:

$$\mathbf{SD}(\, (\mathsf{Ext}(X; S), Z, S)\,,\, (Y, Z, S)\,) \le \epsilon,$$

where S is uniform over $\mathcal{S}$ and Y is uniform over $\mathcal{Y}$.

The following lemma states that universal hash functions are good extractors:

Lemma 2.6 (Leftover Hash Lemma [ILL89, DORS08])**.** *Let $\mathcal{H} = \{H_{\mathsf{seed}} : \mathcal{X} \to \mathcal{Y}\}_{\mathsf{seed}\in\mathcal{S}}$ be a universal hash function family. Then $\mathsf{Ext}(x; \mathsf{seed}) = H_{\mathsf{seed}}(x)$ is a (k, ε)-extractor for $k = \log(|\mathcal{Y}|) + 2\log(1/\varepsilon)$.*

3 Definition of LFE

In this section, we define our notion of laconic function evaluation (LFE) for a class of circuits $\mathcal{C}$. We assume that the class $\mathcal{C}$ associates every circuit $C \in \mathcal{C}$ with some circuit parameters $C.\mathsf{params}$. For our default notion of LFE throughout this paper, unless specified otherwise, we will consider $\mathcal{C}$ to be the class of all circuits with $C.\mathsf{params} = (1^k, 1^d)$ consisting of the input size k and the depth d of the circuit.

Definition 3.1 (LFE)**.** *A laconic function evaluation (LFE) scheme for a class of circuits $\mathcal{C}$ consists of four algorithms* crsGen *,* Compress *,* Enc *and* Dec*.*

- $\mathsf{crsGen}(1^\lambda, \mathsf{params})$ *takes as input the security parameter 1^λ and circuit parameters* params *and outputs a uniformly random* common random string crs *of appropriate length.*[4]
- $\mathsf{Compress}(\mathsf{crs}, C)$ *is a deterministic algorithm that takes as input the common random string* crs *and a circuit $C \in \mathcal{C}$ and outputs a digest* digest_C*.*
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, x)$ *takes as input the common random string* crs*, a digest* digest_C *and a message x and outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct})$ *takes as input the common random string* crs*, a circuit $C \in \mathcal{C}$, and a ciphertext* ct *and outputs a message y.*

We require the following properties from those algorithms:

Correctness: *We require that for all λ,* params *and $C \in \mathcal{C}$ with $C.\mathsf{params} = \mathsf{params}$:*

$$\Pr\left[y = C(x) \;\middle|\; \begin{array}{ccc} \mathsf{crs} & \leftarrow & \mathsf{crsGen}(1^\lambda, \mathsf{params}) \\ \mathsf{digest}_C & = & \mathsf{Compress}(\mathsf{crs}, C) \\ \mathsf{ct} & \leftarrow & \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, x) \\ y & \leftarrow & \mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct}) \end{array} \right] = 1.$$

Security: *We require that there exists a PPT simulator* Sim *such that for all stateful PPT adversary* Adv*, we have:*

$$\left| \Pr\left[\mathrm{EXP}^{Real}_{LFE}(1^\lambda) = 1 \right] - \Pr\left[\mathrm{EXP}^{Ideal}_{LFE}(1^\lambda) \right] \right| \leq \mathsf{negl}(\lambda).$$

for the experiments $\mathrm{EXP}^{Real}_{LFE}(1^\lambda)$ and $\mathrm{EXP}^{Ideal}_{LFE}(1^\lambda)$ defined below:

$\mathrm{EXP}^{Real}_{LFE}(1^\lambda)$:

0. $\mathsf{params} \leftarrow \mathsf{Adv}(1^\lambda)$
1. $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{params})$
2. $x^*, C \leftarrow \mathsf{Adv}(\mathsf{crs})$*:* $C \in \mathcal{C}$, $C.\mathsf{params} = \mathsf{params}$
3. $\mathsf{digest}_C = \mathsf{Compress}(\mathsf{crs}, C)$
4. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, x^*)$
5. *Output* $\mathsf{Adv}(\mathsf{ct})$

$\mathrm{EXP}^{Ideal}_{LFE}(1^\lambda)$:

0. $\mathsf{params} \leftarrow \mathsf{Adv}(1^\lambda)$
1. $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{params})$
2. $x^*, C \leftarrow \mathsf{Adv}(\mathsf{crs})$*:* $C \in \mathcal{C}$, $C.\mathsf{params} = \mathsf{params}$
3. $\mathsf{digest}_C = \mathsf{Compress}(\mathsf{crs}, C)$
4. $\mathsf{ct} \leftarrow \mathsf{Sim}(\mathsf{crs}, C, \mathsf{digest}_C, C(x^*))$
5. *Output* $\mathsf{Adv}(\mathsf{ct})$

We refer to the above as adaptive *security. We also define a weaker version of* selective *security where the above experiments are modified so that* Adv *has to choose x^* at the very beginning of the experiment in step 0, before seeing* crs *(but can still choose C adaptively).*

[4] It would also make sense to allow the crs to be a common *reference* string which is not necessarily uniform. However, since our constructions will have a uniformly random crs we restrict ourselves to this requirement throughout the paper.

Composability. Note that, in our security definition, the simulator is given a correctly generated crs as an input rather than being able to sample it itself. This guarantees composability. Given several ciphertexts ct_i encrypting various inputs x_i under the same or different digests digest_{C_j}, all using the same crs, we can simulate all of them simultaneously and security follows via a simple hybrid argument where we switch them from real to simulated one by one.

Efficiency. The above definition does not directly impose any efficiency restrictions and can therefore be satisfied trivially by setting $\mathsf{digest}_C = C$ and $\mathsf{Enc}(\mathsf{digest}_C, x) = C(x)$. The main goal will be to ensure that the LFE scheme is *laconic*, meaning that the size of $\mathsf{crs}, \mathsf{digest}_C, \mathsf{ct}$ and the run-time Enc should all be as small as possible and certainly smaller than the circuit size of C. We will discuss the efficiency of our constructions as we present them.

4 Construction of LFE from LWE

In this section, we construct LFE for all circuits under the LWE assumption with subexponential modulus-to-noise ratio. As a stepping stone to build LFE, we consider LFE for a restricted class of functionalities, which we call *attribute-based LFE (AB-LFE)* in analogy to attribute-based encryption (ABE).

Definition of AB-LFE. Let $C : \{0,1\}^k \to \{0,1\}$ be a circuit. We define the *Conditional Disclosure Functionality* (CDF) of C as the function

$$\mathsf{CDF}[C](x,\mu) = \begin{cases} (x,\mu) \text{ if } C(x) = 0; \\ (x,\bot) \text{ if } C(x) = 1, \end{cases}$$

where $x \in \{0,1\}^k$, and $\mu \in \{0,1\}$.

We also generalize this to *multi-bit outputs*. For a circuit $C : \{0,1\}^k \to \{0,1\}^\ell$ we define

$$\mathsf{CDF}[C](x,(\mu_1,\ldots,\mu_\ell)) = (x,(\tilde{\mu}_1,\ldots,\tilde{\mu}_\ell)) \text{ where } \begin{cases} \tilde{\mu}_j = \mu_j \text{ if } C_j(x) = 0; \\ \tilde{\mu}_j = \bot \text{ if } C_j(x) = 1. \end{cases},$$

where $x \in \{0,1\}^k, \mu_j \in \{0,1\}^w$ and $C_j(x)$ denotes the j'th output bit of $C(x)$.

Definition 4.1 (AB-LFE)**.** *An AB-LFE for a circuit family $\mathcal{C}$ is an LFE that supports circuits $\mathsf{CDF}[C]$, for all $C \in \mathcal{C}$. We define $\mathsf{CDF}[C].\mathsf{params} = C.\mathsf{params} = (1^k, 1^d)$ where k is the input size and d is the depth of C.*

We will consider canonical descriptions of $\mathsf{CDF}[C]$ such that one can efficiently recover C given $\mathsf{CDF}[C]$. To simplify notation for AB-LFE we will give the algorithms of the AB-LFE the circuit C as input rather than $\mathsf{CDF}[C]$. E.g., we will write $\mathsf{digest}_C \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$ instead of the more cumbersome $\mathsf{digest}_{\mathsf{CDF}[C]} \leftarrow \mathsf{Compress}(\mathsf{crs}, \mathsf{CDF}[C])$.

Outline. In Section 4.1, we build an AB-LFE for circuits C with single-bit output. We then present an optimized construction that supports circuits with multi-bit output in Section 4.2. We discuss adaptive security for our constructions in Section 4.3. In Section 4.4, we give a generic transformation from AB-LFE to LFE (assuming FHE).

4.1 Basic AB-LFE from LWE

We first present our simplest construction of an AB-LFE from LWE for the case of a single bit output.

Parameters. We fix integer parameters $n, m, q, B, B' \in \mathbb{Z}$ and a B-bounded distribution χ over $\mathbb{Z}$, all of which are functions of λ, d. In particular, we set $m = n\lceil \log q \rceil$ and the choice of n, q, χ, B comes from the LWE assumption (see Section 2.2) subject to $n = \mathsf{poly}(\lambda, d), q = 2^{\mathsf{poly}(\lambda,d)}$, and $q/B = 8 \cdot (m+1)^{d+1} \cdot 2^{\lambda}$. We set $B' = B \cdot (m+1)^{d+1} \cdot 2^{\lambda}$.

Construction. We define the procedures crsGen , $\mathsf{Compress}$, Enc and Dec as follows.

- $\mathsf{crsGen}(1^{\lambda}, \mathsf{params} = (1^k, 1^d))$: Pick k random matrices $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n \times m}$ and set:
$$\mathsf{crs} = (\mathbf{A}_1, \ldots, \mathbf{A}_k).$$
- $\mathsf{Compress}(\mathsf{crs}, C)$: Output:
$$\mathsf{digest}_C = \mathbf{A}_C = \mathsf{EvalPK}(C, \{\mathbf{A}_i\}_{i \le k}),$$
where EvalPK is defined in Section 2.3.
- $\mathsf{Enc}(\mathsf{crs}, \mathbf{A}_C, (\mathbf{x}, \mu))$: Pick a random $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_i \leftarrow \chi^m$ for $i \le k$, and compute:
$$\mathbf{b}_i = \mathbf{s}^{\top}(\mathbf{A}_i - x_i \mathbf{G}) + \mathbf{e}_i^{\top}, \quad i \le k.$$
Sample $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\$)$,[5] set $\mathbf{d} = \mathbf{A}_C \cdot \mathbf{t}$, and compute
$$\beta = \mathbf{s}^{\top}\mathbf{d} + \widetilde{e} + \mu \cdot \lfloor q/2 \rceil,$$
where $\widetilde{e} \leftarrow [-B', B']$. Output:
$$\mathsf{ct} = (\{\mathbf{b}_i\}_{i \le k},\ \beta,\ \mathbf{t},\ x).$$
- $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct})$: If $C(x) = 1$ output $\perp$. Else compute:
$$\widetilde{\mathbf{b}} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i \le k}, \{\mathbf{b}_i\}_{i \le k}, x),$$
where EvalCT is defined in Section 2.3. Output $\lfloor (\beta - \widetilde{\mathbf{b}}^{\top}\mathbf{t})/q \rceil$.

Proof overview. The key observation underlying both correctness and security is that for honestly generated ciphertexts, we have:
$$\widetilde{\mathbf{b}} \approx \mathbf{s}^{\top}[\mathbf{A}_C - C(x)\mathbf{G}]$$
and therefore
$$\beta - \widetilde{\mathbf{b}} \cdot \mathbf{t} \approx C(x) \cdot \mathbf{s}^{\top}\mathbf{G} \cdot \mathbf{t}. \tag{2}$$
In the proof, we will use the above equation to simulate β as $\widetilde{\mathbf{b}} \cdot \mathbf{t} + C(x) \cdot \mathbf{s}^{\top}\mathbf{G} \cdot \mathbf{t}$.

Claim 4.2 (Correctness)**.** *The construction is correct.*

Proof. Assume $C(x) = 0$. Then, by correctness of EvalCT, we have:
$$\widetilde{\mathbf{b}} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i \le k}, \{\mathbf{b}_i\}_{i \le k}, x) = \mathbf{s}^{\top}[\mathbf{A}_C - C(x)\mathbf{G}] + \mathbf{e}_C^{\top} = \mathbf{s}^{\top}\mathbf{A}_C + \mathbf{e}_C^{\top},$$
with $\|\mathbf{e}_C\|_{\infty} \le (m+1)^d \cdot B$. Therefore $\widetilde{\mathbf{b}}^{\top}\mathbf{t} = \mathbf{s}^{\top}\mathbf{d} + \mathbf{e}_C^{\top} \cdot \mathbf{t}$ and
$$\beta - \widetilde{\mathbf{b}}^{\top}\mathbf{t} = (\mathbf{s}^{\top}\mathbf{d} + \widetilde{e} + \mu \cdot \lfloor q/2 \rceil) - (\mathbf{s}^{\top}\mathbf{d} + \mathbf{e}_C^{\top} \cdot \mathbf{t}) = \mu \cdot \lfloor q/2 \rceil + \widetilde{e} - \mathbf{e}_C^{\top} \cdot \mathbf{t}$$
where $|\widetilde{e} - \mathbf{e}_C^{\top} \cdot \mathbf{t}| \le (m+1)^{d+1} \cdot B + B' < 2B \cdot (m+1)^{d+1} \cdot 2^{\lambda} < q/4$. □

[5] Recall $\mathbf{G}^{-1}(\$)$ denotes the distribution of sampling $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ uniformly at random and outputting $\mathbf{t} = \mathbf{G}^{-1}(\mathbf{u})$.

Claim 4.3 (Security)**.** *The construction is selectively secure under the LWE assumption $LWE_{n,q,\chi}$.*

Proof. Note that the simulator is given C, x^* and $\mathsf{CDF}[C](x^*, \mu^*)$ as input. In particular, if $C(x^*) = 0$ then the simulator is given (x^*, μ^*) in full and can therefore create the ciphertext honestly as in the Real experiment. So without loss of generality, we can concentrate on the case $C(x^*) = 1$. Define the following simulator:

- $\mathsf{Sim}(\mathsf{crs}, \mathsf{digest}_C, C, x^*)$: pick $\mathbf{b}_i \leftarrow \mathbb{Z}_q^m$ for $i \le k$, pick $\beta \leftarrow \mathbb{Z}_q$, and $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\$)$. Output:

$$\mathsf{ct} = (\{\mathbf{b}_i\}_{i\le k},\, \beta,\, \mathbf{t},\, x^*).$$

We prove that the experiments $\text{EXP}_{LFE}^{Real}(1^\lambda)$ and $\text{EXP}_{LFE}^{Ideal}(1^\lambda)$ from Definition 3.1 are computationally indistinguishable via a sequence of hybrids.

Hybrid 0. This is the real experiment $\text{EXP}_{LFE}^{Real}(1^\lambda)$.

Hybrid 1. Here the way β is computed is modified. After computing $\mathbf{b}_i = \mathbf{s}^\top(\mathbf{A}_i - x_i^*\mathbf{G}) + \mathbf{e}_i^\top$ and sampling $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\$)$, compute $\widetilde{\mathbf{b}} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i\le k}, \{\mathbf{b}_i\}_{i\le k}, x^*)$ and $\alpha = \mathbf{s}^\top\mathbf{G}\mathbf{t} + e_0 \in \mathbb{Z}_q$, where $e_0 \leftarrow \chi$. Set

$$\beta = \widetilde{\mathbf{b}} \cdot \mathbf{t} + \alpha + \widetilde{e} + \mu \cdot \lfloor q/2 \rceil,$$

where $\widetilde{e} \leftarrow [-B', B']$.

By correctness of EvalCT and the fact that $C(x^*) = 1$, we have $\widetilde{\mathbf{b}} = \mathbf{s}^\top(\mathbf{A}_C - \mathbf{G}) + \mathbf{e}_C^\top$, so that

$$\widetilde{\mathbf{b}} \cdot \mathbf{t} + \alpha = \mathbf{s}^\top\mathbf{A}_C \cdot \mathbf{t} + \mathbf{e}_C^\top \cdot \mathbf{t} + e_0 = \mathbf{s}^\top\mathbf{d} + \mathbf{e}_C^\top \cdot \mathbf{t} + e_0$$

and therefore, in Hybrid 1, we have

$$\beta = \mathbf{s}^\top\mathbf{d} + \mu^* \cdot \lfloor q/2 \rceil + \widetilde{e} + (\mathbf{e}_C^\top \cdot \mathbf{t} + e_0)$$

Then, by Lemma 2.3, the distributions of $\widetilde{e} + \mathbf{e}_C^\top\mathbf{t} + e_0$ and $\widetilde{e}$ are statistically close; and therefore the distribution of β in Hybrid 1 is statistically close to that in Hybrid 0.

Hybrid 2. In this hybrid we pick α and $\{\mathbf{b}_i\}_{i\le k}$ uniformly at random. We still sample $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\$)$ and compute $\widetilde{\mathbf{b}} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i\le k}, \{\mathbf{b}_i\}_{i\le k}, x)$ and $\beta = \widetilde{\mathbf{b}} \cdot \mathbf{t} + \alpha + \widetilde{e} + \mu^* \cdot \lfloor q/2 \rceil$ as previously.

We show that Hybrid 1 and Hybrid 2 are computationally indistinguishable under $\text{LWE}_{n,q,\chi}$. More precisely, the reduction receives x^* from the adversary and gets LWE challenges $(\mathbf{u}, \alpha)$ and $(\mathbf{M_i}, \mathbf{b}_i)_{i\le k}$, where $\mathbf{u} \leftarrow \mathbb{Z}_q^n, \mathbf{M_i} \leftarrow \mathbb{Z}_q^{n\times m}$. It sets $\mathsf{crs} = \{\mathbf{A}_i = \mathbf{M_i} + x_i^*\mathbf{G}\}_{i\le k}$, and $\mathbf{t} = \mathbf{G}^{-1}(\mathbf{u})$, and computes β as previously. Then, if α and $\{\mathbf{b}_i\}$ are distributed as LWE samples, then the view of the adversary is as in Hybrid 1; if they are uniform, then its view is distributed as in Hybrid 2.

Note that Hybrid 2 corresponds to $\text{EXP}_{LFE}^{Ideal}(1^\lambda)$ with the simulator Sim defined above since the values β and $\{\mathbf{b}_i\}_{i\le k}$ in Hybrid 2 are uniformly random. □

Comparison with the ABE from [BGG$^+$14]. Our construction is very reminiscent of the ABE from [BGG$^+$14]. In their ABE scheme, there is an additional matrix $\mathbf{A}_0$, and the relation $\mathbf{d} = \mathbf{A}_C \cdot \mathbf{t}$ is replaced with $\mathbf{d} = [\mathbf{A}_0 \mid \mathbf{A}_C] \cdot \mathbf{t}$. The quantities $\mathbf{A}_0, \mathbf{d}$ are part of the master public key and $\mathbf{t}$ is the secret key corresponding to the circuit C. The ABE security proof needs to take into account an adversary that sees many such $\mathbf{t}$'s with respect to the same $\mathbf{d}$.

There are several key differences between our AB-LFE and the prior ABE scheme. A crucial distinction is that our Encryption algorithm takes $\mathbf{A}_C$ as input, which allows it to sample $\mathbf{t}$ and then set $\mathbf{d} = \mathbf{A}_C \cdot \mathbf{t}$.

The value $\mathbf{t}$ is a part of our ciphertext. In [BGG+14], the analogous operation is performed by having a trapdoor for $\mathbf{A}_0$ as a master secret key of the ABE and using this trapdoor to sample $\mathbf{t}$ that satisfies $\mathbf{d} = [\mathbf{A}_0 \mid \mathbf{A}_C] \cdot \mathbf{t}$ during key generation and including it as part of the secret key for the circuit C. The proof of security for the ABE scheme is also significantly more complex and involves carefully "puncturing" the public matrices in $\mathbf{A}_i$ in the crs so as to be able to provide secret keys for circuits C if and only if $C(x) = 1$. The fact that we don't need any trapdoors for our construction essentially paves our way to a much simpler proof.

Message-adaptivity. To argue indistinguishability between Hybrids 1 and 2 from LWE, the reduction needs to know the challenge attribute x^* ahead of time to create the crs using its LWE challenges, which makes the construction selectively secure. However, it doesn't need to know the message μ^* at that point. In particular, our construction is "*message-adaptive*", where the adversary has to choose the challenge attribute x^* before seeing the crs, but can choose the challenge message μ^* *after* seeing the crs.

Efficiency. For circuits with input length k, 1-bit output and depth d the above construction of AB-LFE from LWE has the following parameters.

- The crs is of size $k \cdot \mathsf{poly}(\lambda, d)$. The digest is of size $\mathsf{poly}(\lambda, d)$.
- The run-time of the encryption algorithm and the size of the ciphertext are $k \cdot \mathsf{poly}(\lambda, d)$.
- The run-time of the compression and the decryption algorithms is $|C| \cdot \mathsf{poly}(\lambda, d)$.

4.2 Improved AB-LFE for Multi-Bit Output

We extend the construction to AB-LFE with multi-bit output. Recall that in this case the encryption algorithm takes as input $(x, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\ell)$ and the decryption algorithm recovers $\boldsymbol{\mu}_j$ for all j such that the j'th bit of $C(x)$ is 0. We use the notation $C_j(x)$ to denote the j'th bit of $C(x)$. As another difference from the single-bit case, we now also allow the messages $\boldsymbol{\mu}_j \in \{0,1\}^w$ to be multi-bit messages rather than a single bit.

A naive solution to achieve AB-LFE with ℓ-bit output is to use ℓ separate AB-LFE's with single-bit output for the circuits C_j and to encrypt (x, μ_j) separately under each one (we can reuse the same crs across all of them). In that case, all efficiency measures blow up by a multiplicative factor of ℓ: the size of the ciphertext would be $\ell \cdot k \cdot \mathsf{poly}(\lambda, d)$, the size of the digest would be $\ell \cdot \mathsf{poly}(\lambda, d)$ and the run-time of the compression/decryption algorithms would be $\ell \cdot |C| \cdot \mathsf{poly}(\lambda, d)$. We show that we can do better in two steps. First, we show how to compress the ciphertext size to only $(\ell + k) \cdot \mathsf{poly}(\lambda, d)$ instead of $\ell \cdot k \cdot \mathsf{poly}(\lambda, d)$ and the encryption/decryption run-time to $|C| \cdot \mathsf{poly}(\lambda, d)$ instead of $\ell \cdot |C| \cdot \mathsf{poly}(\lambda, d)$. In essence, we show how to reuse the "x part" of the ciphertext across all ℓ copies. Second, we show how to also compress the digest from $\ell \cdot \mathsf{poly}(\lambda, d)$ to just $\mathsf{poly}(\lambda)$ without blowing up any of the other parameters.

4.2.1 Compressing the Ciphertext

Construction. We fix the parameters n, m, q, B, B', χ as in the single bit case.

- $\mathsf{crsGen}(1^\lambda, \mathsf{params} = (1^k, 1^d))$: Pick k random matrices $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n \times m}$ and set:
$$\mathsf{crs} = (\mathbf{A}_1, \ldots, \mathbf{A}_k).$$
- $\mathsf{Compress}(\mathsf{crs}, C)$: Let $C : \{0,1\}^k \to \{0,1\}^\ell$. Output:
$$\mathsf{digest}_C = \{\mathbf{A}_{C_j}\}_{j \le \ell} = \mathsf{EvalPK}(C, \{\mathbf{A}_i\}_{i \le k}),$$
where EvalPK over multi-bit output circuits is defined in Section 2.3.

- $\mathsf{Enc}(\mathsf{crs}, \{\mathbf{A}_{C_j}\}_{j\le\ell}, (x, \{\boldsymbol{\mu}_j\}_{j\le\ell}))$: Let $\boldsymbol{\mu}_j \in \{0,1\}^w$ for $j \le \ell$.
  Pick $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_i \leftarrow \chi^m$ for $i \le k$, and compute:

$$\mathbf{b}_i = \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G}) + \mathbf{e}_i^\top, \quad i \le k,$$

  For $j \le \ell$, sample $\mathbf{T}_j \leftarrow (\mathbf{G}^{-1}(\$))^w \in \mathbb{Z}_q^{m\times w}$, set $\mathbf{D}_j = \mathbf{A}_{C_j} \cdot \mathbf{T}_j$, and compute for all $j \le \ell$

$$\boldsymbol{\beta}_j = \mathbf{s}^\top\mathbf{D}_j + \widetilde{\mathbf{e}}_j^\top + \boldsymbol{\mu}_j \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q^{1\times w},$$

  where $\widetilde{\mathbf{e}}_j \leftarrow [-B, B]^w$. Output:

$$\mathsf{ct} = (\{\mathbf{b}_i\}_{i\le k}, \{\boldsymbol{\beta}_j\}_{j\le\ell}, \{\mathbf{T}_j\}_{j\le\ell}, x).$$

- $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct})$: Compute

$$\left\{\widetilde{\mathbf{b}}_j\right\}_{j\le\ell} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i\le k}, \{\mathbf{b}_i\}_{i\le k}, x).$$

  For all $j \le \ell$, if $C_j(x) = 1$ set $\boldsymbol{\mu}_j = \perp$; else set $\boldsymbol{\mu}_j = \lfloor(\boldsymbol{\beta}_j - \widetilde{\mathbf{b}}_j\mathbf{T}_j)/q\rceil$. Output $(\boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_\ell)$.

Correctness follows directly by the same argument as for Claim 4.2. For security, we want to ensure that $\boldsymbol{\mu}_j$ is hidden whenever $C_j(x^*) = 1$. However, there are some differences from the single output bit case here since we may have $C_j(x^*) = 1$ for some j and $C_j(x^*) = 0$ for others.

Claim 4.4 (Security)**.** *The above construction is selectively secure under the LWE assumption $LWE_{n,q,\chi}$.*

Proof. Note that the simulator is given $\mathsf{CDF}[C](x^*, (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\ell))$ as input. Denote by I the set of indices $j \le \ell$ such that $C_j(x^*) = 0$. For all $j \in I$, the simulator is given $\boldsymbol{\mu}_j$ while for all other $j \notin I$ the simulator does not get $\boldsymbol{\mu}_j$. Define the following new simulator:

- $\mathsf{Sim}(\mathsf{crs}, \mathsf{digest}_C, C, \mathsf{CDF}[C](x^*, (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\ell)))$: Pick $\mathbf{b}_i \leftarrow \mathbb{Z}_q^m$ for all $i \le k$ and $\mathbf{T}_j \leftarrow (\mathbf{G}^{-1}(\$))^w$ for all $j \le \ell$, and compute:

$$\left\{\widetilde{\mathbf{b}}_j\right\}_{j\le\ell} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i\le k}, \{\mathbf{b}_i\}_{i\le k}, x^*).$$

  - For all $j \in I$ compute:
$$\boldsymbol{\beta}_j = \widetilde{\mathbf{b}}_j \cdot \mathbf{T}_j + \widetilde{\mathbf{e}}_j^\top + \boldsymbol{\mu}_j \cdot \lfloor q/2 \rceil,$$
    where $\widetilde{\mathbf{e}}_j \leftarrow [-B', B']^w$.
  - For all $j \notin I$ pick $\boldsymbol{\beta}_j \leftarrow \mathbb{Z}_q$ uniformly at random.

  Output:
$$\mathsf{ct} = (\{\mathbf{b}_i\}_{i\le k}, \{\boldsymbol{\beta}_j\}_{j\le\ell}, \{\mathbf{T}_j\}_{j\le\ell}, x^*).$$

The hybrids follow closely the ones of the proof for the 1-bit version in Section 4.1. There are a few differences:

Hybrid 1. The way the vectors $\boldsymbol{\beta}_j$ are generated now depends on $C_j(x^*)$.
We compute $\mathbf{b}_i = \mathbf{s}^\top(\mathbf{A}_i - x_i^*\mathbf{G}) + \mathbf{e}_i^\top$ for $i \le k$ (where $\mathbf{e}_i \leftarrow \chi^m$), and pick $\mathbf{T}_j \leftarrow (\mathbf{G}^{-1}(\$))^w$ for all $j \le \ell$, and set $\left\{\widetilde{\mathbf{b}}_j\right\}_{j\le\ell} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i\le k}, \{\mathbf{b}_i\}_{i\le k}, x^*)$. Then, for all $j \le \ell$:

- If $j \in I$ (i.e. $C_j(x^*) = 0$), we directly set $\boldsymbol{\beta}_j = \widetilde{\mathbf{b}}_j \cdot \mathbf{T}_j + \widetilde{\mathbf{e}}_j^\top + \boldsymbol{\mu}_j \cdot \lfloor q/2 \rceil$, where $\widetilde{\mathbf{e}}_j \leftarrow [-B', B']^w$.

- If $j \notin I$ (i.e. $C_j(x^*) = 1$) we compute $\boldsymbol{\alpha}_j = \mathbf{s}^\top \mathbf{G}\mathbf{T}_j + \mathbf{e}_{0,j}^\top$, where $\mathbf{e}_{0,j} \leftarrow \chi^w$, and set $\boldsymbol{\beta}_j = \widetilde{\mathbf{b}}_j \cdot \mathbf{T}_j + \boldsymbol{\alpha}_j + \widetilde{\mathbf{e}}_j^\top + \boldsymbol{\mu}_j \cdot \lfloor q/2 \rceil$, where $\widetilde{\mathbf{e}}_j \leftarrow [-B', B']^w$.

Indistinguishability from the Hybrid 0 follows by the same argument as in the previous proof, as $\widetilde{\mathbf{b}}_j = \mathbf{s}^\top[\mathbf{A}_{C_j} - C_j(x^*)\mathbf{G}] + \mathbf{e}_{C_j}^\top$ by correctness of EvalCT, where $C_j(x^*) = 0$ if $j \in I$ and $C_j(x^*) = 1$ otherwise. Lemma 2.3 then ensures that the distributions of $\widetilde{\mathbf{e}}_j^\top$ (noise of $\boldsymbol{\beta}_j$ in Hybrid 0), and $\widetilde{\mathbf{e}}_j^\top + \mathbf{e}_{C_j}^\top \mathbf{T}_j + \mathbf{e}_{0,j}^\top$ (noise in Hybrid 1 if $C_j(x^*) = 0$)) and $\widetilde{\mathbf{e}}_j^\top + \mathbf{e}_{C_j}^\top \mathbf{T}_j$ (noise in Hybrid 1 if $C_j(x^*) = 1$) are statistically close; and therefore the distributions of $\boldsymbol{\beta}_j$ in both hybrids are statistically indistinguishable.

Hybrid 2. In this hybrid we pick $\mathbf{b}_i$ at random for all $i \leq k$, as well as the $\boldsymbol{\alpha}_j$ for all $j \notin I$. We still sample $\mathbf{T}_j \leftarrow (\mathbf{G}^{-1}(\$))^w$ and compute $\left\{\widetilde{\mathbf{b}}_j\right\}_{j \leq \ell} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i \leq k}, \{\mathbf{b}_i\}_{i \leq k}, x)$ and $\boldsymbol{\beta}_j$ according to $C_j(x^*)$ as previously.

We show that Hybrid 1 and Hybrid 2 are computationally indistinguishable under $\mathrm{LWE}_{n,q,\chi}$. More precisely, a Reduction receives $\{\mathbf{M}_i, \mathbf{b}_i\}_{i \leq k}$ for all $i \leq k$ and $\{\mathbf{U}_j, \boldsymbol{\alpha}_j\}_{j \notin I}$ from the LWE challenge (where $\mathbf{U}_j \in \mathbb{Z}_q^{n \times w}$). It sets $\mathsf{crs} = \{\mathbf{A}_i = \mathbf{M}_i + x_i^* \mathbf{G}\}$, samples $\mathbf{T}_j \leftarrow (\mathbf{G}^{-1}(\$))^w$ and computes $\boldsymbol{\beta}_j$ as previously. If $\{\boldsymbol{\alpha}_j\}_{j \notin I}$ and $\{\mathbf{b}_i\}$ are LWE samples, then the view of the Adversary is as in Hybrid 1; if they are uniform, its view is distributed as in Hybrid 2.

Now Hybrid 2 corresponds to $\mathrm{EXP}_{LFE}^{Ideal}(1^\lambda)$ with the simulator Sim defined above, which concludes the proof. □

Efficiency. For circuits $C : \{0,1\}^k \to \{0,1\}^\ell$ of depth d and for message-length w the above construction of AB-LFE from LWE has the following parameters:

- The CRS is of size $k \cdot \mathsf{poly}(\lambda, d)$. The digest is of size $\ell \cdot \mathsf{poly}(\lambda, d)$.
- The run-time of the encryption algorithm and the size of the ciphertext are $(k + \ell \cdot w) \cdot \mathsf{poly}(\lambda, d)$.
- The run-time of the compression algorithm is $|C| \cdot \mathsf{poly}(\lambda, d)$. The run-time of the decryption algorithm is $(|C| + \ell \cdot w) \cdot \mathsf{poly}(\lambda, d)$.

4.2.2 Compressing the Digest

We now show how to update the previous construction further to reduce the digest size from "small" $\ell \cdot \mathsf{poly}(\lambda, d)$ to "tiny" $\mathsf{poly}(\lambda)$. We do this generically using *Laconic OT (LOT)* [CDG+17]. Instead of giving the original "small" digest, we compress it further into a "tiny" digest using LOT. The encryptor then creates a garbled circuit that takes as input the small digest digest_C and computes the ciphertext under the previous construction. The new ciphertext consists of the garbled circuit and LOT encryptions of the garbled circuit labels.

Laconic OT. A Laconic OT [CDG+17] is an LFE scheme for functions of the form $f_D(i, \mu_0, \mu_1) = (i, \mu_{D[i]})$ where $D[i]$ denotes the i'th bit of D. We write $\mathsf{LOT.Compress}(\mathsf{crs}, D)$ and shorthand for $\mathsf{LOT.Compress}(\mathsf{crs}, f_D)$. Note that such LOT can be seen as a special form of AB-LFE by running it twice; once with the function $f_{0,D}(i) = D[i]$ and the attribute/message pair (i, μ_0) and once with the function $f_{1,D}(i) = 1 - D[i]$ and the attribute/message pair (i, μ_1). However, for LOT we require an additional efficiency requirement:

- The compression algorithm $\mathsf{LOT.Compress}(\mathsf{crs}, D)$ outputs the digest digest_D along with a processed database $\hat{D}$.

- The decryption algorithm $\mathsf{Dec}^{\hat{D}}(\mathsf{crs}, \mathsf{ct})$ runs in time $\mathsf{poly}(\lambda, \log |D|)$ given RAM access to the processed database $\hat{D}$. Moreover, for any index i, there is a circuit of size $\mathsf{poly}(\lambda, \log |D|)$ that is given $\mathsf{crs}, \mathsf{ct}$ and some subset of the bits of $\hat{D}$ that depend only on i, and outputs $\mathsf{Dec}^{\hat{D}}(\mathsf{crs}, \mathsf{ct})$.
- The crs and the digest are of size $\mathsf{poly}(\lambda)$, the compression run-time is $|D|\mathsf{poly}(\lambda, \log |D|)$ and the encryption run-time is $\mathsf{poly}(\lambda, \log |D|)$.

The work of [CDG+17] shows that one can take a "mildly compressing" LOT which has no efficiency requirements other than that the digest is at most 1/2 the size of the database and bootstrap it to construct an LOT with the above efficiency. Since such mildly compressing LOT immediately follows from AB-LFE, this gives us a construction of LOT under the LWE assumption (the above is overkill and there are simpler direct constructions of mildly compressing LOT from LWE that don't go through AB-LFE as was noted in e.g. [BLSV18] but never fully specified). We also have LOT constructions under many standard assumptions such as CDH, Factoring and even LPN with extremely low noise [CDG+17, DG17, BLSV18].

Construction. Let $\mathsf{LOT} = (\mathsf{LOT.crsGen}, \mathsf{LOT.Compress}, \mathsf{LOT.Enc}, \mathsf{LOT.Dec})$ be an LOT scheme as defined above. Let $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ be a circuit garbling scheme as in Section 2.5. Let $\mathsf{LFE} = (\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ be the AB-LFE. We construct an AB-LFE scheme $\mathsf{LFE}' = (\mathsf{crsGen}', \mathsf{Compress}', \mathsf{Enc}', \mathsf{Dec}')$ with a compressed digest as follows.

- $\mathsf{crsGen}'(1^\lambda, \mathsf{params})$: Run $\mathsf{crs}_{\mathsf{LOT}} \leftarrow \mathsf{LOT.crsGen}(1^\lambda), \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{params})$. Output $\mathsf{crs}' = (\mathsf{crs}_{\mathsf{LOT}}, \mathsf{crs})$.
- $\mathsf{Compress}'(\mathsf{crs}', C)$: Run $\mathsf{digest}_C \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$. Run $(\mathsf{digest}', \hat{D}) \leftarrow \mathsf{LOT.Compress}(\mathsf{crs}_{\mathsf{LOT}}, \mathsf{digest}_C)$. Output digest'.
- $\mathsf{Enc}'(\mathsf{crs}', (x, \{\boldsymbol{\mu}_j\}_{j\le\ell}))$. Let $E(\cdot)$ be the circuit that has $\mathsf{crs}, (x, \{\boldsymbol{\mu}_j\}_{j\le\ell})$ and randomness r hard-coded, takes as input digest_C and outputs $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, (x, \{\boldsymbol{\mu}_j\}_{j\le\ell}); r)$. Let $(\Gamma, \{L_i^0, L_i^1\}_{i\in[t]}) \leftarrow \mathsf{GC.Garble}(1^\lambda, E)$ be the garbled circuit and labels. Let $\{\mathsf{ct}_i \leftarrow \mathsf{LOT.Enc}(\mathsf{crs}_{\mathsf{LOT}}, (i, L_i^0, L_i^1))\}_{i\in[t]}$. Output $\mathsf{ct}' = (\Gamma, \{\mathsf{ct}_i\}_{i\in[t]})$.
- $\mathsf{Dec}'(\mathsf{crs}', C, \mathsf{ct}')$: Run $\mathsf{digest}_C \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$ and $(\mathsf{digest}', \hat{D}) \leftarrow \mathsf{LOT.Compress}(\mathsf{crs}_{\mathsf{LOT}}, \mathsf{digest}_C)$. Decrypt $L_i = \mathsf{LOT.Dec}^{\hat{D}}(\mathsf{crs}, \mathsf{ct}_i)$. Compute $\mathsf{ct} = \mathsf{GC.Eval}(\Gamma, \{L_i\}_{i\le k})$. Output $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct})$.

The correctness of the above construction follows immediately.

Claim 4.5. *If* LFE *is a (selectively) secure AB-LFE,* GC *is a circuit garbling scheme and* LOT *is a secure* LOT *then* LFE' *is a (selectively) secure AB-LFE.*

Proof. The simulator $\mathsf{Sim}'(\mathsf{crs}', \mathsf{digest}', C, y)$ for LFE' first runs the simulator $\mathsf{Sim}(\mathsf{crs}, \mathsf{digest}_C, C, y)$ of LFE to get a ciphertext ct. It then runs the simulator of the garbled circuit $\mathsf{Sim}_{GC}(1^\lambda, \mathsf{ct})$ with the output ct to get a simulated garbled circuit and input $(\Gamma, \{L_i\}_{i\in[t]})$. Lastly it runs the LOT simulator $\mathsf{LOT.Sim}(\mathsf{crs}_{\mathsf{LOT}}, \mathsf{digest}', (i, L_i))$ to get ciphertexts ct_i for $i \in [t]$. It outputs $\mathsf{ct}' = (\Gamma, \{\mathsf{ct}_i\}_{i\in[t]})$.

We rely on a hybrid argument to show the indistinguishability of the real world and the simulation.

Hybrid 0. This is the real experiment $\text{EXP}_{LFE}^{Real}(1^\lambda)$.

Hybrid 1. In this experiment, during the computation of the challenge ciphertext, instead of choosing $\{\mathsf{ct}_i \leftarrow \mathsf{LOT.Enc}(\mathsf{crs}_{\mathsf{LOT}}, (i, L_i^0, L_i^1))\}_{i\in[t]}$ we set $\mathsf{ct}_i \leftarrow \mathsf{LOT.Sim}(\mathsf{crs}_{\mathsf{LOT}}, \mathsf{digest}', (i, L_i))$ where $L_i = L_i^{b_i}$ where b_i is the i'th bit of digest_C.

Hybrids 0 and 1 are indistinguishable by the security of the LOT scheme.

Hybrid 2. In this experiment, during the computation of the challenge ciphertext, instead of choosing $(\Gamma, \{L_i^0, L_i^1\}_{i\in[t]}) \leftarrow \mathsf{GC.Garble}(1^\lambda, E)$ we choose $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, (x, \{\boldsymbol{\mu}_j\}_{j\le\ell}))$ and set $(\Gamma, \{L_i\}_{i\in[t]}) \leftarrow \mathsf{Sim}_{GC}(1^\lambda, \mathsf{ct})$.

Hybrids 1 and 2 are indistinguishable by the security of the circuit garbling scheme.

Hybrid 3. In this experiment, during the computation of the challenge ciphertext, instead of choosing $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, (x, \{\boldsymbol{\mu}_j\}_{j\le\ell}))$ we set $\mathsf{ct} \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{digest}_C, C, y)$ where $y = \mathsf{CDF}[C](x, \{\boldsymbol{\mu}_j\}_{j\le\ell})$.

Hybrids 2 and 3 are indistinguishable by the security of LFE.

Hybrid 3 corresponds to $\text{EXP}_{LFE}^{Ideal}(1^\lambda)$ with the simulator Sim' defined above, which concludes the proof. □

Efficiency. In the above construction, if we let LFE be the AB-LFE scheme from Section 4.2.1, then we get the following parameters:

- The size of the CRS is $k \cdot \mathsf{poly}(\lambda, d)$. The digest is of size $\mathsf{poly}(\lambda)$.
- The run-time of the encryption algorithm and the size of the ciphertext are $\widetilde{O}(k + \ell \cdot w) \cdot \mathsf{poly}(\lambda, d)$.
- The run-time of the compression algorithm is $|C| \cdot \mathsf{poly}(\lambda, d)$. The run-time of the decryption algorithm is $\widetilde{O}(|C| + \ell \cdot w) \cdot \mathsf{poly}(\lambda, d)$.

4.3 Adaptive AB-LFE Security from Adaptive LWE

Our constructions of AB-LFE were only proven to be selectively secure under LWE. We now show how to prove adaptive security by relying on a new but natural LWE assumption where we give the adversary some very limited choice over the coefficients used to create LWE samples. In more detail, the adversary is given matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$, adaptively chooses $x \in \{0,1\}^k$ and gets LWE samples with the coefficients $\mathbf{A}_i - x_i\mathbf{G}$. We also give the adversary arbitrarily many additional LWE samples with random coefficients.

Definition 4.6 ((Decision) Adaptive LWE)**.** *We define the decision adaptive LWE assumption $ALWE_{n,k,q,\chi}$ with parameter $n, k, q \in \mathbb{Z}$ and a distribution χ over $\mathbb{Z}$ which are all parametrized by the security parameter λ. Let $m = n\lceil \log q \rceil$. We let $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ be the gadget matrix (see Section 2.3).*[6] *For any polynomial $m' = m'(\lambda)$ we consider the following two games $\mathbf{Game}^\beta$ with $\beta \in \{0,1\}$ between a challenger and an adversary $\mathcal{A}$.*

- *The Challenger picks k random matrices $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n\times m}$ for $i \le k$, and sends them to $\mathcal{A}$.*
- *$\mathcal{A}$ adaptively picks $x \in \{0,1\}^k$, and sends it to the Challenger.*
- *The Challenger samples $\mathbf{s} \leftarrow \mathbb{Z}_q^n$. It computes for all $i \le k$:*

$$\begin{cases} \mathbf{b}_i = \mathbf{s}^\top(\mathbf{A}_i - x_i \cdot \mathbf{G}) + \mathbf{e}_i^\top \text{ where } \mathbf{e}_i \leftarrow \chi^m & \text{if } \beta = 0, \\ \mathbf{b}_i \leftarrow \mathbb{Z}_q^m & \text{if } \beta = 1. \end{cases}$$

It also picks $\mathbf{A}_{k+1} \leftarrow \mathbb{Z}_q^{n\times m'}$ and computes

$$\begin{cases} \mathbf{b}_{k+1} = \mathbf{s}^\top \mathbf{A}_{k+1} + \mathbf{e}_{k+1}^\top \text{ where } \mathbf{e}_{k+1} \leftarrow \chi^{m'} & \text{if } \beta = 0, \\ \mathbf{b}_{k+1} \leftarrow \mathbb{Z}_q^{m'} & \text{if } \beta = 1. \end{cases}$$

It sends $\mathbf{A}_{k+1}$, $\{\mathbf{b}_i\}_{i\le k+1}$ to $\mathcal{A}$.

[6] The assumption would be meaningful for any other choice of matrix $\mathbf{G}$ as well.

The $ALWE_{n,k,q,\chi}$ assumption states that for all polynomial $m = m(\lambda)$ the games $\mathbf{Game}^0$ *and* $\mathbf{Game}^1$ *are computationally indistinguishable.*

Adaptive LWE implies Adaptive AB-LFE. We notice that the constructions of AB-LFE in Sections 4.1 and 4.2 are adaptively secure under the adaptive $ALWE_{n,k,q,\chi}$ assumption where k is the input size of the circuit. This follows directly from the proofs of indistinguishability - in particular, the indistinguishability of Hybrids 1 and 2 which relied on the LWE assumption. (The additional LWE challenge values $\mathbf{A}_{k+1}, \mathbf{b}_{k+1}$ with non-adaptively selected coefficients are used to create $(\mathbf{u}, \alpha)$ in the single bit case or $\{\mathbf{U}_j, \boldsymbol{\alpha}_j\}_{j\in[\ell]}$ in the multi-bit case needed to generate the ciphertext.)

Security of Adaptive LWE. Note that if the adversary was forced to choose x before seeing $\{\mathbf{A}_i\}$ then the security of ALWE would follow directly from LWE. We also have a reduction from $LWE_{n,q,\chi}$ to $ALWE_{n,k,q,\chi}$ with exponential security loss 2^k, where the reduction *guesses* in advance the adaptive choice x that the adversary makes during the interaction with the challenger. Therefore, under the sub-exponential security of LWE we can show the security of ALWE but all of the parameters (lattice dimension, modulus size etc.) have to scale polynomially with k. Nevertheless, it seems plausible to assume that ALWE has a much higher level of security than this reduction implies and that the parameters do not have to scale polynomially with k. We summarize the above by outlining two possible parameter settings for ALWE.

- *Optimistic Parameters:* For any polynomial $p = p(\lambda)$ there exists some polynomial $n = n(\lambda)$, some $q = q(\lambda) = 2^{\mathsf{poly}(\lambda)}$ and some $B = B(\lambda)$-bounded distribution $\chi = \chi(\lambda)$ such that $q/B \geq 2^p$ and the $ALWE_{n,k,q,\chi}$ assumption holds for all polynomial $k = k(\lambda)$.
- *Provable Parameters:* For any polynomials $p = p(\lambda), k = k(\lambda)$ there exists some polynomial $n = n(\lambda)$, some $q = q(\lambda) = 2^{\mathsf{poly}(\lambda)}$ and some $B = B(\lambda)$-bounded distribution $\chi = \chi(\lambda)$ such that $q/B \geq 2^p$ and the $LWE_{n,k,q,\chi}$ assumption holds.

Note the difference – with the provable parameters, the choice of n, q, χ can depend on k while with the optimistic parameters they do not. The ALWE assumption with provable parameters follows from the sub-exponential security of LWE. The ALWE assumption with optimistic parameters is plausibly secure but we do not have any meaningful reduction from LWE.

Efficiency of Adaptive AB-LFE. To summarize, under the ALWE assumption with optimistic parameters we get an adaptively secure AB-LFE with the same parameters as the selectively secure schemes in Section 4.2. Under the ALWE assumption with provable parameters, which follows from the sub-exponentially secure LWE assumption, we get an adaptively secure AB-LFE where:

- The crs is of size $\mathsf{poly}(\lambda, k, d)$. The digest is of size $\mathsf{poly}(\lambda)$.
- The run-time of the encryption algorithm and the size of the ciphertext is $\ell \cdot w \cdot \mathsf{poly}(\lambda, k, d)$.
- The run-time of the compression algorithm is $|C| \cdot \mathsf{poly}(\lambda, k, d)$.
- The run-time of the decryption algorithm is $(|C| + \ell \cdot w) \cdot \mathsf{poly}(\lambda, k, d)$.

4.4 From AB-LFE to LFE via FHE

We construct a compiler which converts AB-LFE to LFE, assuming the existence of any (leveled) Fully Homomorphic Encryption (FHE) scheme (defined in Section 2.4). In particular, combined with our construction of an AB-LFE under LWE and leveled FHE under LWE (e.g., [GSW13]), we get an LFE under LWE.

Our compiler is very similar to the one introduced in [GKP+13], which compiles any ABE into a single key FE (assuming FHE). The distinction between ABE and FE is analogous to the one between AB-LFE and LFE. Intuitively, the compiler works as follows. Alice creates an AB-LFE digest for the circuit $\mathsf{FHE.Eval}(C, \cdot)$ which takes as input an encryption $\widehat{x}$ of a value x and outputs an encryption $\widehat{y}$ of $y = C(x)$. The encryptor (Bob) first uses an FHE scheme to encrypt his input x, resulting in an FHE ciphertext $\widehat{x}$. He then garbles the circuit $\mathsf{FHE.Dec}_{\mathsf{sk}}$, which has the FHE secret key sk hard-coded inside of it and performs a decryption. Lastly, he uses the AB-LFE scheme with an attribute $\widehat{x}$ and the labels of the garbled circuit as messages so that Alice recovers exactly the labels that correspond to the homomorphically evaluated ciphertext $\widehat{y} = \mathsf{FHE.Eval}(C, \widehat{x})$. Bob sends the garbled circuit along with the AB-LFE ciphertext to Alice as his LFE ciphertext. Alice uses AB-LFE decryption to recover the labels of $\widehat{y}$ and then feeds these to the garbled circuit to recover y in the clear.

Notation: Two-Outcome AB-LFE. We introduce a piece of simplifying notation to simplify our description of the compiler. In standard AB-LFE, the encryption algorithm gets $(x, \mu_1, \ldots, \mu_\ell)$ and the decryption recovers the messages μ_j for all j such that $C_j(x) = 0$ and does not learn anything about the others. In "two-outcome ABE" the encryption algorithm gets $(x, \{(\mu_j^0, \mu_j^1)\}_{j \le \ell})$ and the decryption algorithm recovers $\mu_j^{C_j(x)}$. Given an AB-LFE scheme (crsGen , $\mathsf{Compress}$, Enc and Dec) we think of "two-outcome AB-LFE" (crsGen , $\mathsf{Compress}'$, Enc' and Dec') as syntactic sugar for the following:

- $\mathsf{Compress}'(\mathsf{crs}, C)$: Given $C : \{0,1\}^k \to \{0,1\}^\ell$, define $\widetilde{C} : x \mapsto (C(x) \| \overline{C}(x))$, where $\overline{C}(x)$ is the bitwise complement of $C(x)$. Output $\mathsf{Compress}(\mathsf{crs}, \widetilde{C})$.
- $\mathsf{Enc}'(\mathsf{crs}, \mathsf{digest}_C, \Big(x, \{(\mu_j^0, \mu_j^1)\}_{j \le \ell}\Big))$: Output $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, (x, \mu_1^0, \ldots, \mu_\ell^0, \mu_1^1, \ldots, \mu_\ell^1))$.
- $\mathsf{Dec}'(\mathsf{crs}, C, \mathsf{ct})$: Let $(\tilde{\mu}_1^0, \ldots, \tilde{\mu}_\ell^0, \tilde{\mu}_1^1, \ldots, \tilde{\mu}_\ell^1) = \mathsf{Dec}(\mathsf{crs}, \widetilde{C}, \mathsf{ct})$. Set μ_i to be the one of $\tilde{\mu}_i^0, \tilde{\mu}_i^1$ which is not $\perp$ and output $\mu_1, \ldots, \mu_\ell$.

Construction. Let $(\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ be a leveled FHE (defined in Section 2.4), and $(\mathsf{GC.Garble}, \mathsf{GC.Eval})$ be a garbling scheme (defined in Section 2.5). Let $(\mathsf{AB\text{-}LFE.crsGen}, \mathsf{AB\text{-}LFE.Compress}, \mathsf{AB\text{-}LFE.Enc}, \mathsf{AB\text{-}LFE.Dec})$ be a "two-outcome" AB-LFE as explained above. We assume the leveled FHE is such that if C is a circuit of depth d then $\mathsf{FHE.Eval}(C, \mathsf{ct})$ can be computed by a circuit of depth $d'(\lambda, d) = \mathsf{poly}(\lambda, d)$ and if $x \in \{0,1\}^k$ then the encryption of x is of size $k'(\lambda, k, d) = k \cdot \mathsf{poly}(\lambda, d)$.

Our LFE scheme is defined as follows.

- $\mathsf{crsGen}(1^\lambda, \mathsf{params} = (1^k, 1^d))$: Output: $\mathsf{crs} \leftarrow \mathsf{AB\text{-}LFE.crsGen}(1^\lambda, (1^{k'(\lambda,k,d)}, 1^{d'(\lambda,d)}))$.
- $\mathsf{Compress}(\mathsf{crs}, C)$: Output:
$$\mathsf{digest}_C = \mathsf{AB\text{-}LFE.Compress}(\mathsf{crs}, \mathsf{FHE.Eval}(C, \cdot)).$$
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, x)$: Generate keys for the homomorphic scheme $(\mathsf{hpk}, \mathsf{hsk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$ and compute $\widehat{x} = \mathsf{FHE.Enc}_{\mathsf{hpk}}(x)$.
  Compute: $(\Gamma, \{L_j^0, L_j^1\}_{j \le \ell}) \leftarrow \mathsf{GC.Garble}(\mathsf{FHE.Dec}_{\mathsf{hsk}}(\cdot))$.
  Compute: $\mathsf{ct}_{\mathsf{AB\text{-}LFE}} \leftarrow \mathsf{AB\text{-}LFE.Enc}\Big(\mathsf{crs}, \mathsf{digest}_C, \widehat{x}, \{L_j^0, L_j^1\}_{j \le \ell}\Big)$.
  Output: $\mathsf{ct} = (\Gamma, \mathsf{ct}_{\mathsf{AB\text{-}LFE}})$.
- $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct})$: Let $\{L_j\}_{j \le \ell} = \mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, \mathsf{FHE.Eval}(C, \cdot), \mathsf{ct}_{\mathsf{AB\text{-}LFE}})$. Output:
$$\mu = \mathsf{GC.Eval}(\Gamma, \{L_j\}_{j \le \ell}).$$

Claim 4.7. *Assuming correctness of the underlying AB-LFE, FHE and garbling scheme, the construction above is correct.*

Proof. By correctness of the underlying "two-outcome" AB-LFE, in the Decryption algorithm, $\mathsf{AB\text{-}LFE.Dec}$ recovers $L_j = L_j^{\mathsf{FHE.Eval}_j(C,\widehat{x})}$, where $\mathsf{FHE.Eval}_j(C,\widehat{x})$ denotes the j'th bit of $\mathsf{FHE.Eval}(C,\widehat{x})$. Then by correctness of the garbling scheme, $\mathsf{GC.Eval}$ outputs $\mathsf{FHE.Dec}_{\mathsf{hsk}}(\mathsf{FHE.Eval}(C,\widehat{x}))$, which is $C(x)$ by correctness of the FHE. □

Claim 4.8. *Assuming the underlying AB-LFE is selectively (resp. adaptively) secure, and the security of the FHE and garbling scheme, the construction above is selectively (resp. adaptively) secure.*

Proof. The proof is very similar to the one in [GKP+13], Section 3.2.

Define the following simulator:

- $\mathsf{Sim}(\mathsf{crs}, \mathsf{digest}_C, C, C(x^*))$: Pick $(\mathsf{hpk}, \mathsf{hsk}) \leftarrow \mathsf{FHE.KeyGen}()$, and compute $\widehat{0} \leftarrow \mathsf{FHE.Enc}_{\mathsf{hpk}}(0)$. Run the simulator for the garbling scheme $\mathsf{Sim}_{GC}(|\mathsf{FHE.Eval}(C,\cdot)|, |\widehat{0}|, C(x^*))$ to get $(\widetilde{\Gamma}, \{\widetilde{L}_j\}_{j\leq\ell})$, where $|\mathsf{FHE.Eval}(C,\cdot)|$ is efficiently computable given C.

  Run the AB-LFE Simulator $\mathsf{Sim}_{\mathsf{AB\text{-}LFE}}$ to obtain:

$$\mathsf{ct}_{\mathsf{AB\text{-}LFE}} \leftarrow \mathsf{Sim}_{\mathsf{AB\text{-}LFE}}(\mathsf{crs}, \mathsf{digest}_C, \mathsf{FHE.Eval}(C,\cdot), \left(\widehat{0}, \{\widetilde{L}_j\}_{j\leq\ell}\right)).$$

  Output:

$$\mathsf{ct} = (\widetilde{\Gamma}, \mathsf{ct}_{\mathsf{AB\text{-}LFE}}).$$

We prove the claim via a sequence of hybrids.

Hybrid 0. This is the real experiment $\text{EXP}_{LFE}^{Real}(1^\lambda)$.

Hybrid 1. The way $\mathsf{ct}_{\mathsf{AB\text{-}LFE}}$ is generated is modified. Let $d_j = \mathsf{FHE.Eval}_j(C,\widehat{x^*})$ be the j'th bit of $\mathsf{FHE.Eval}(C,\widehat{x^*})$. Compute:

$$\mathsf{ct}_{\mathsf{AB\text{-}LFE}} \leftarrow \mathsf{Sim}_{\mathsf{AB\text{-}LFE}}(\mathsf{crs}, \mathsf{digest}_C, \mathsf{FHE.Eval}(C,\cdot), \left(\widehat{x^*}, \{L_j^{d_j}\}_{j\leq\ell}\right)).$$

We have that Hybrid 0 is computationally indistinguishable from Hybrid 1 by selective (resp. adaptive) security of the underlying AB-LFE. This is because for all j, the message $\mu_j^{d_j} = L_j^{d_j}$ is authorized, while $\mu_j^{\overline{d_j}} = L_j^{\overline{d_j}}$ is not. In particular, the AB-LFE Simulator $\mathsf{Sim}_{\mathsf{AB\text{-}LFE}}$ can simulate $\mathsf{ct}_{\mathsf{AB\text{-}LFE}}$ given only $L_j^{d_j}$ (and not $L_j^{\overline{d_j}}$).

Hybrid 2. The way $\widetilde{\Gamma}$ and $\mathsf{ct}_{\mathsf{AB\text{-}LFE}}$ are generated is modified. We now use the garbling scheme simulator Sim_{GC} (defined in Section 2.5) to compute $(\widetilde{\Gamma}, \{\widetilde{L}_j\}_{j\leq\ell}) \leftarrow \mathsf{Sim}_{GC}(C(x^*), |\mathsf{FHE.Eval}(C,\cdot)|)$, where $|\mathsf{FHE.Eval}(C,\cdot)|$ can be computed using C. Compute now:

$$\mathsf{ct}_{\mathsf{AB\text{-}LFE}} \leftarrow \mathsf{Sim}_{\mathsf{AB\text{-}LFE}}(\mathsf{crs}, \mathsf{digest}_C, \mathsf{FHE.Eval}(C,\cdot), \widehat{x^*}, \{\widetilde{L}_j\}_{j\leq\ell}),$$

and output $\mathsf{ct} = (\widetilde{\Gamma}, \mathsf{ct}_{\mathsf{AB\text{-}LFE}})$.

This is indistinguishable from Hybrid 1 by security of the garbling scheme (defined in Section 2.5), as, by correctness of the FHE, we have $\mathsf{FHE.Dec}(\mathsf{FHE.Eval}(C,\widehat{x^*})) = C(x^*)$.

Hybrid 3. This the ideal experiment $\text{EXP}_{LFE}^{Ideal}(1^\lambda)$ with the simulator Sim defined as above, i.e. $\widehat{x^*}$ is now replaced by $\widehat{0} = \mathsf{FHE.Enc}_{\mathsf{hpk}}(0)$, so that:

$$\mathsf{ct}_{\mathsf{AB\text{-}LFE}} \leftarrow \mathsf{Sim}_{\mathsf{AB\text{-}LFE}}(\mathsf{crs}, \mathsf{digest}_C, \mathsf{FHE.Eval}(C,\cdot), \left(\widehat{0}, \{\widetilde{L}_j\}_{j\leq\ell}\right)),$$

This is indistinguishable from Hybrid 2 by security of the FHE scheme. □

4.5 Summary of LFE Construction and Parameters

Combining our construction of AB-LFE and our compiler from AB-LFE to LFE we get the following results.

Selective. Under the LWE assumption with subexponential modulus-to-noise ratio, there exists a selectively secure LFE scheme for circuits C of depth d, input size k and output size ℓ where:

- The crs is of size $k \cdot \mathsf{poly}(\lambda, d)$. The digest is of size $\mathsf{poly}(\lambda)$.
- The run-time of the encryption algorithm and the size of the ciphertext are $\widetilde{O}(k + \ell) \cdot \mathsf{poly}(\lambda, d)$.
- The run-time of the compression and the decryption algorithms is $\widetilde{O}(|C|) \cdot \mathsf{poly}(\lambda, d)$.

Adaptive. Under the adaptive LWE (ALWE) assumption with "optimistic parameters" there exists an adaptive LFE scheme with the same efficiency as the selective scheme above.

Under the ALWE assumption with "provable parameters" which follows from the sub-exponential security of LWE there exists an adaptive LFE scheme where:

- The crs is of size $\mathsf{poly}(\lambda, k, d)$. The digest is of size $\mathsf{poly}(\lambda)$.
- The run-time of the encryption algorithm and the size of the ciphertext are $\widetilde{O}(\ell) \cdot \mathsf{poly}(\lambda, k, d)$.
- The run-time of the compression and the decryption algorithms is $\widetilde{O}(|C|) \cdot \mathsf{poly}(\lambda, k, d)$.

5 Function-Hiding LFE

In this section, we study an additional security property for LFEs, that we call *Function Hiding*. In particular, we want the digest $\mathsf{digest}_C = \mathsf{Compress}(\mathsf{crs}, C)$ to completely hide the circuit C. Note that we cannot achieve any reasonable security when the algorithm $\mathsf{Compress}$ is *deterministic*. We will therefore consider *randomized* compression algorithms $\mathsf{Compress}$ when considering function hiding. The decryptor needs to remember the randomness of the compression algorithm as a *secret key*, which is needed to decrypt LFE ciphertexts created under the resulting digest.

5.1 Overview of Results

We first provide a definition of function hiding LFEs in Section 5.2. Then in Section 5.3 we provide a generic construction of a statistical function hiding LFE, given any standard LFE. In Appendix A, we also give an alternate more direct construction of statistical function hiding LFE by modifying our LWE based construction.

Overview of the Generic Construction. Our generic construction proceeds in two steps.

In a first step (Section 5.3.1), we show how to upgrade any LFE to function-hiding LFE using generic 2-round 2PC (e.g., based on oblivious transfer (OT) and garbled circuits). This is essentially the same idea used to get receiver privacy in LOT [CDG+17]. Instead of Alice sending the digest digest_C to Bob, and Bob computing $\mathsf{Enc}(\mathsf{digest}_C, x)$, we use a 2-round 2PC protocol where Alice has input digest_C, Bob has input x, and Alice learns $\mathsf{Enc}(\mathsf{digest}_C, x)$. We set the new digest of the function-hiding scheme to be Alice's first message in the 2PC and we set the new encryption of the function-hiding scheme to be Bob's reply in the 2PC.[7] The efficiency only degrades by some fixed $\mathsf{poly}(\lambda)$ multiplicative factors. If we rely on an OT scheme where receiver privacy holds statistically, we get a statistical function hiding security. If we use a 2-round 2PC based on OT and garbled circuits then the transformation preserver

[7] We need to rely on semi-malicious secure 2PC where security holds even if Alice chooses her randomness maliciously.

the asymptotic efficiency of the construction – the digest size and the ciphertext size only grow by some fixed $\mathsf{poly}(\lambda)$ multiplicative factor.

In the second step (Section 5.3.2), we show how to generically construct (semi-malicious secure) OT with statistical receiver security in the CRS model from AB-LFE (without function hiding). In fact, the work of [BLSV18] already shows that laconic OT (LOT) implies public-key encryption and the result there can easily be strengthened to show that it implies such OT as well.[8] Since LOT is a special case of AB-LFE our result follows. However, for completeness, we give a somewhat simpler construction of OT from AB-LFE. The CRS of the OT scheme consists of a CRS for an AB-LFE scheme as well as an extractor seed seed. The OT receiver chooses a sufficiently long random value r and encrypts his bit b by setting $\hat{b} = b \oplus \mathsf{Ext}(r; \mathsf{seed})$. In addition the receiver creates an AB-LFE digest for the function C_r which takes as input $\mathsf{seed}, \hat{b}$, and decrypts it to get $b = \hat{b} \oplus \mathsf{Ext}(r; \mathsf{seed})$. The receiver message consists of the LFE digest and the bit $\hat{b}$. To argue that the receiver message statistically hides the receiver bit b, we rely on the fact that the digest short and we think of it as some leakage on r which is independent of seed; therefore, by extractor security, b is statistically hidden even given $\hat{b}$ and the digest. The sender uses the AB-LFE to encrypt the message μ_0 using the attribute $x = (\mathsf{seed}, \hat{b})$ and the message μ_1 using the attribute $x = (\mathsf{seed}, 1 \oplus \hat{b})$ to ensure that the receiver can decrypt only μ_b.

Overview of the Direct Construction. Our direct construction makes a small modification to our LWE-based AB-LFE scheme from Sections 4.1, 4.2 to directly add function-hiding security. In that scheme, the digest was $\mathbf{A}_C = \mathsf{EvalPK}(C, \{\mathbf{A}_i\}_{i \le k})$ where the matrices $\mathbf{A}_i$ are in the CRS. To get function hiding, we add some additional matrices $\{\mathbf{B}_j\}_{j \in [N]}$ to the CRS and the compression algorithm picks a random $r \leftarrow \{0,1\}^N$ and sets the digest to be $\widetilde{\mathbf{A}}_C = \mathbf{A}_C + \sum_{j=1}^{N} r_j \mathbf{B}_j$. The Leftover Hash Lemma (Lemma 2.6) states that when N is sufficiently large then the new digest is statistically close to uniform. In some sense we can think of $\widetilde{\mathbf{A}}_C$ as a digest for the function $C'(x, z) = C(x) + \langle r, z \rangle$. To encrypt under an attribute x, the encryptor then uses the AB-LFE scheme with an attribute $(x, z = 0^N)$. This introduces essentially no loss in parameters.

The above gives us a statistically function-hiding AB-LFE under LWE. It is easy to see that our compiler from AB-LFE to LFE from Section 4.4 preserves function hiding and therefore we get a statistically function hiding LFE under LWE.

5.2 Definition of Function-Hiding LFE

Firstly, we modify the syntax of LFE as follows. We will consider a *randomized* compression function $\mathsf{Compress}(\mathsf{crs}, C; r)$ which uses random coins r, and now the decryption $\mathsf{Dec}(\mathsf{crs}, C, r, \mathsf{ct})$ needs the same randomness r to decrypt. We modify the definitions of correctness and encryption security in Definition 3.1 to match this change of notation as follows:

- Correctness states that for any r, if $\mathsf{digest}_C = \mathsf{Compress}(\mathsf{crs}, C; r)$, and if $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, x)$, then $\mathsf{Dec}(\mathsf{crs}, f, r, \mathsf{ct})$ outputs $C(x)$ with probability 1.
- For encryption security, we modify the experiments $\text{EXP}^{Real}_{LFE}(1^\lambda)$ and $\text{EXP}^{Ideal}_{LFE}(1^\lambda)$ (defined in Definition 3.1) as follows:
  - In step 2 the adversary now chooses x^*, C, r.
  - In step 3 we set the digest to $\mathsf{digest}_C = \mathsf{Compress}(\mathsf{crs}, C; r)$.
  - In step 4 of $\text{EXP}^{Ideal}_{LFE}(1^\lambda)$ we now also give the simulator the randomness r and set $\mathsf{ct} \leftarrow \mathsf{Sim}(\mathsf{crs}, (C, r), \mathsf{digest}_C, C(x^*))$.

[8] Personal communication from Alex Lombardi.

In the selective security version, the adversary chooses x^* ahead of time in step 0, but can still choose C, r in step 2 after seeing the crs. The fact that we allow the adversary to choose the randomness r corresponds to a semi-malicious Alice that chooses adversarial randomness in computing the digest but we still want to guarantee security for Bob's ciphertext.

Finally, we add the following function-hiding security requirement:

Definition 5.1 ((Statistical) Function-Hiding)**.** *An LFE is* function-hiding *if there exists a PPT simulator* Sim_{FH} *such that for all stateful PPT adversary* Adv*, we have:*

$$\left| \Pr\left[\mathrm{EXP}_{FH}^{Real}(1^\lambda) = 1 \right] - \Pr\left[\mathrm{EXP}_{FH}^{Ideal}(1^\lambda) \right] \right| \leq \mathsf{negl}(\lambda).$$

for the experiments defined below:

$\mathrm{EXP}_{FH}^{Real}(1^\lambda)$:

0. $\mathsf{params} \leftarrow \mathsf{Adv}(1^\lambda)$
1. $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{params})$
2. $C \leftarrow \mathsf{Adv}(\mathsf{crs})$*:*
   $C \in \mathcal{C}$, $C.\mathsf{params} = \mathsf{params}$
3. $\mathsf{digest}_C \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$
4. *Output* $\mathsf{Adv}(\mathsf{digest}_C)$

$\mathrm{EXP}_{FH}^{Ideal}(1^\lambda)$:

0. $\mathsf{params} \leftarrow \mathsf{Adv}(1^\lambda)$
1. $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{params})$
2. $C \leftarrow \mathsf{Adv}(\mathsf{crs})$*:*
   $C \in \mathcal{C}$, $C.\mathsf{params} = \mathsf{params}$
3. $\mathsf{digest}_C \leftarrow \mathsf{Sim}_{FH}(\mathsf{crs}, C.\mathsf{params})$.
4. *Output* $\mathsf{Adv}(\mathsf{digest}_C)$

We say that the scheme is statistically *function hiding if the above holds for all (even inefficient) adversaries* Adv*.*

5.3 Generic Construction

We now show how to generically convert any LFE scheme (which is sufficiently compressing) into a function-hiding LFE with essentially the same asymptotic efficiency.

5.3.1 From 2-round OT to Statistical Function Hiding

We first show that any LFE, combined with any 2-round OT with statistical receiver security, implies statistical Function Hiding LFE.

Recall: 2-round OT. Our syntax and definition of OT is close to the one from [BL18].

Let $(\mathsf{OT.crsGen}, \mathsf{OT.Query}, \mathsf{OT.Response}, \mathsf{OT.Output})$ be a 2-round OT with the following syntax:

- $\mathsf{crs} \leftarrow \mathsf{OT.crsGen}(1^\lambda)$ outputs a crs.
- $M^{(1)} = \mathsf{OT.Query}(\mathsf{crs}, b; r)$ computes the first message $M^{(1)}$ of the OT from the receiver to the sender where b is the receiver's choice bit and r denotes the receiver's private random coins.
- $M^{(2)} \leftarrow \mathsf{OT.Response}(\mathsf{crs}, M^{(1)}, \mu_0, \mu_1)$ computes the second message $M^{(2)}$ from the sender to the receiver, where μ_0, μ_1 are the sender's messages.
- $\mu_b = \mathsf{OT.Output}(M^{(2)}, b, r)$ decrypts the sender message $M^{(2)}$ using the receiver's private random coins r.

We will suppose that the OT is *statistically* secure with respect to the Receiver, and computationally secure with respect to the sender. Moreover the sender's security holds even for a semi-malicious receiver. This is defined formally below.

- **Statistical Receiver security**: We require that the two following distributions are statistically close:

$$(\mathsf{crs}, \mathsf{OT.Query}(\mathsf{crs}, 0)) \stackrel{\mathrm{s}}{\approx} (\mathsf{crs}, \mathsf{OT.Query}(\mathsf{crs}, 1)));$$

where $\mathsf{crs} \leftarrow \mathsf{OT.crsGen}(1^\lambda)$.

- **Sender security (with Semi-Malicious Receiver)**: We require that for all stateful PPT adversary Adv, we have:

$$\left| \Pr\left[\mathrm{EXP}^0_{OT}(1^\lambda) = 1 \right] - \Pr\left[\mathrm{EXP}^1_{OT}(1^\lambda) \right] \right| \leq \mathsf{negl}(\lambda).$$

for the experiment defined below:

$\mathrm{EXP}^\rho_{OT}(1^\lambda)$:

1. $\mathsf{crs} \leftarrow \mathsf{OT.crsGen}(1^\lambda)$
2. $(r, b, \mu_0, \mu_1) \leftarrow \mathsf{Adv}(\mathsf{crs})$
3. $M^{(1)} = \mathsf{OT.Query}(\mathsf{crs}, b; r)$,
   If $\rho = 0$ then $M^{(2)} \leftarrow \mathsf{OT.Response}(\mathsf{crs}, M^{(1)}, \mu_0, \mu_1)$
   else $M^{(2)} \leftarrow \mathsf{OT.Response}(\mathsf{crs}, M^{(1)}, \mu_b, \mu_b)$.
4. Output $\mathsf{Adv}(M^{(2)})$.

Construction of Function-Hiding LFE. Let $(\mathsf{LFE.crsGen}, \mathsf{LFE.Compress}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$ be an LFE with deterministic compression. We define the function-hiding LFE via the following algorithms $(\mathsf{FH.crsGen}, \mathsf{FH.Compress}, \mathsf{FH.Enc}, \mathsf{FH.Dec})$:

- $\mathsf{FH.crsGen}(1^\lambda, \mathsf{params})$: Output $\mathsf{crs} = (\mathsf{LFE.crs} \leftarrow \mathsf{LFE.crsGen}(1^\lambda, \mathsf{params}), \mathsf{OT.crs} \leftarrow \mathsf{OT.crsGen}(1^\lambda))$.
- $\mathsf{FH.Compress}(\mathsf{crs}, C; r)$: Compute $\mathsf{LFE.digest} = \mathsf{LFE.Compress}(\mathsf{LFE.crs}, C)$.
  Let $k = |\mathsf{LFE.digest}|$. Output:

$$\mathsf{FH.digest}_C = \{\mathsf{OT.Query}\left(\mathsf{OT.crs}, (\mathsf{LFE.digest})_i\,;\, r_i\right)\}_{i \leq k}$$

  where $(\mathsf{LFE.digest})_i$ denotes the ith bit of $\mathsf{LFE.digest}$ and $r = \{r_i\}_{i \leq k}$ denotes the randomness used for the first message of the OT.

- $\mathsf{FH.Enc}(\mathsf{crs}, \mathsf{FH.digest}_C, x)$: Sample some randomness t and compute:

$$(\Gamma, \{L^b_i\}_{i \leq k}) \leftarrow \mathsf{GC.Garble}\left(\mathsf{LFE.Enc}(\mathsf{LFE.crs}, \cdot, x\,;\, t)\right),$$

  (where $\mathsf{LFE.Enc}(\mathsf{LFE.crs}, \cdot, x\,;\, t)$ takes as input a digest $\mathsf{LFE.digest}$ and outputs an LFE ciphertext using randomness t).

  Parse $\mathsf{FH.digest}_C$ as $\{M^{(1)}_i\}_{i \leq k}$, and output:

$$\mathsf{FH.ct} = \left(\Gamma, \left\{\mathsf{OT.Response}(\mathsf{OT.crs}, M^{(1)}_i, L^0_i, L^1_i)\right\}_{i \leq k}\right).$$

- $\mathsf{FH.Dec}(\mathsf{crs}, C, r = \{r_i\}_{i \leq k}, \mathsf{FH.ct} = (\Gamma, \{M^{(2)}_i\}_{i \leq k}))$: Compute $\mathsf{LFE.digest} = \mathsf{LFE.Compress}(\mathsf{LFE.crs}, C)$.
  For all $i \leq k$, recover from the OT the ith label

$$L_i = \mathsf{OT.Output}(M^{(2)}_i, (\mathsf{LFE.digest})_i, r_i).$$

Compute:

$$\mathsf{LFE.ct} = \mathsf{GC.Eval}(\Gamma, L_i),$$

and output:

$$x' = \mathsf{LFE.Dec}(\mathsf{LFE.crs}, C, \mathsf{LFE.ct}).$$

Claim 5.2 (Correctness)**.** *The LFE* $(\mathsf{FH.crsGen}, \mathsf{FH.Compress}, \mathsf{FH.Enc}, \mathsf{FH.Dec})$ *is correct.*

Proof. By correctness of the OT, we have:

$$L_i = \mathsf{OT.Output}(M_i^{(2)}, (\mathsf{LFE.digest})_i, r_i) = L_i^{(\mathsf{LFE.digest})_i} \text{ for all } i \leq k.$$

Then, by correctness of the garbling scheme:

$$\mathsf{LFE.ct} = \mathsf{GC.Eval}(\Gamma, L_i) = \mathsf{LFE.Enc}(\mathsf{LFE.crs}, \mathsf{LFE.digest}, x\,;\, t).$$

Finally, by correctness of the underlying LFE, we have:

$$x' = \mathsf{LFE.Dec}(\mathsf{LFE.crs}, C, \mathsf{LFE.ct}) = C(x).$$

□

Claim 5.3 (Security)**.** *Suppose that the OT satisfies sender security and that* LFE *is selectively (resp. adaptively) secure. Then* FH *is selectively (resp. adaptively) secure.*

Proof. Define the following Simulator Sim for the LFE experiment:

- $\mathsf{Sim}(\mathsf{FH.crs}, \mathsf{FH.digest}_C, C, C(x^*))$:

  Let LFE.Sim be a simulator for the underlying LFE scheme (given by LFE security), and GC.Sim be a simulator for the garbled circuit (given by security of the garbling scheme).

  Compute $\mathsf{LFE.digest} = \mathsf{LFE.Compress}(\mathsf{LFE.crs}, C)$, and let:

  $$\mathsf{LFE.ct}_{\mathsf{Sim}} \leftarrow \mathsf{LFE.Sim}(\mathsf{LFE.crs}, \mathsf{LFE.digest}, C, C(x^*)).$$

  Compute:

  $$(\widetilde{\Gamma}, \{\widetilde{L}_i\}_{i \leq k}) \leftarrow \mathsf{GC.Sim}(\mathsf{LFE.ct}_{\mathsf{Sim}}, |\mathsf{LFE.Enc}|),$$

  Parse $\mathsf{FH.digest}_C$ as $\{M_i^{(1)}\}_{i \leq k}$ and output:

  $$\mathsf{FH.ct} = \left(\widetilde{\Gamma}, \left\{\mathsf{OT.Response}(\mathsf{OT.crs}, M_i^{(1)}, \widetilde{L}_i, \widetilde{L}_i)\right\}_{i \leq k}\right).$$

Hybrid 0. This is the real experiment, as defined in Definition 3.1.

Hybrid 1. Now, given C, compute $\mathsf{LFE.digest} = \mathsf{LFE.Compress}(\mathsf{LFE.crs}, C)$, and let $b_i = (\mathsf{LFE.digest})_i$ be the ith choice bit computed in FH.Compress.

Now the second part of the ciphertext is generated as:

$$\mathsf{OT.Response}(\mathsf{OT.crs}, M_i^{(1)}, L_i^{b_i}, L_i^{b_i}), \quad \text{for all } i \leq k.$$

This is indistinguishable from Hybrid 0 by sender security of the OT.

Hybrid 2. Now compute $\mathsf{LFE.ct} = \mathsf{LFE.Enc}(\mathsf{LFE.crs}, \mathsf{LFE.digest}, x)$; and set $(\widetilde{\Gamma}, \{\widetilde{L}_i\}_{i\leq k}) \leftarrow \mathsf{GC.Sim}(\mathsf{LFE.ct}, N)$. The ciphertext is now:

$$\mathsf{FH.ct} = \left(\widetilde{\Gamma}, \left\{\mathsf{OT.Response}(\mathsf{OT.crs}, M_i^{(1)}, \widetilde{L}_i, \widetilde{L}_i)\right\}_{i\leq k}\right).$$

This is indistinguishable from Hybrid 1 by the security of the garbling scheme.

Hybrid 3. Now compute $\mathsf{LFE.digest} = \mathsf{LFE.Compress}(\mathsf{LFE.crs}, C)$, and set:

$$\mathsf{LFE.ct}_{\mathsf{Sim}} \leftarrow \mathsf{LFE.Sim}(\mathsf{LFE.crs}, \mathsf{LFE.digest}, C, C(x^*)),$$

where $\mathsf{LFE.Sim}$ is the simulator given by security of the underlying LFE.

Now generate: $(\widetilde{\Gamma}, \{\widetilde{L}_i\}_{i\leq k}) \leftarrow \mathsf{GC.Sim}(\mathsf{LFE.ct}_{\mathsf{Sim}}, N)$.

In particular, this corresponds to the ideal experiment with the simulator Sim defined above.

This is indistinguishable from Hybrid 2 by security of the underlying LFE. In particular, if the initial LFE is selectively (resp. adaptively) secure, then $(\mathsf{FH.crsGen}, \mathsf{FH.Compress}, \mathsf{FH.Enc}, \mathsf{FH.Enc})$ is selectively (resp. adaptively) secure.

□

Claim 5.4 (Statistical Function Hiding)**.** *Suppose the OT achieves statistical receiver security. Then the LFE scheme* FH *is statistically function hiding.*

Proof. Define the following Simulator $\mathsf{Sim}_{\mathsf{FH}}$ for the Function Hiding experiment:

- $\mathsf{Sim}_{\mathsf{FH}}(\mathsf{crs}, \mathsf{params})$: Output

$$\mathsf{FH.digest}_C = \{\mathsf{OT.Query}(\mathsf{OT.crs}, 0)\}_{i\leq k}.$$

The statistical function-hiding security of FH with the above simulator follows immediately from the statistical receiver security of the OT scheme via k hybrids, where we switch the i'th OT query from $\mathsf{OT.Query}(\mathsf{OT.crs}, (\mathsf{LFE.digest})_i)$ to $\mathsf{OT.Query}(\mathsf{OT.crs}, 0)$.

□

5.3.2 AB-LFE implies 2-round OT

We now construct a 2-round OT with statistical receiver security from any selectively secure AB-LFE (without function hiding) which is sufficiently compressing.

On a high level, a naive way to obtain an OT from an LFE would be for the receiver to compute a digest of the circuit $C_b : (\mu_0, \mu_1) \mapsto \mu_b$ using her choice bit b and for the sender to encrypt the message pair (μ_0, μ_1). However, to get receiver security we would need the LFE to already be function hiding but our goal is to construct OT from an LFE scheme which does not yet have function hiding security. We notice that if the LFE is sufficiently compressing then it must provide some limited form of function hiding since the digest is too short to reveal the function in its entirety. We use this observation to mask the OT receiver bit b by compressing a much a larger circuit with a large amount of randomness.

Construction. Let $N = N(\lambda)$ be an integer to be determined later, and let $\mathsf{Ext} : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}$ be a *strong extractor* (see Section 2.6) given by $\mathsf{Ext}(r;\mathsf{seed}) = \langle r, \mathsf{seed}\rangle$. By the Leftover Hash Lemma (Lemma 2.6), this is a (k,ε)-extractor for any entropy $k = \omega(\log\lambda)$ with $\varepsilon = \mathsf{negl}(\lambda)$. Notice that $\mathsf{Ext}(r;\mathsf{seed})$ can be computed with a circuit of depth $\mathcal{O}(\log N)$.

For $r \in \{0,1\}^N$, define the following circuit C_r:

$$C_r(\mathsf{seed}, \hat{b}) = \hat{b} \oplus \mathsf{Ext}(r;\mathsf{seed}).$$

Let $(\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ be an AB-LFE scheme which is "sufficiently compressing" meaning the following: for a sufficiently large $N(\lambda) = \mathsf{poly}(\lambda)$ the size of $\mathsf{digest} = \mathsf{Compress}(\mathsf{LFE.crs}, C_r)$ is $|\mathsf{digest}| \le N(\lambda) - \lambda$. Note that this is a very mild compression requirement and clearly achieved our schemes from Section 4. We set $N = N(\lambda)$ to be this sufficiently large polynomial.

Define the following 2-round OT. We assume that the message μ_0, μ_1 are single bits. To get OT for larger message size we can compose single bit OTs together.

- $\mathsf{OT.crsGen}(1^\lambda)$: Output $\mathsf{OT.crs} = (\mathsf{LFE.crs}, \mathsf{seed})$ where $\mathsf{seed} \leftarrow \{0,1\}^N$ and $\mathsf{LFE.crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{params})$ where $\mathsf{params} = (1^k, 1^d)$ corresponds to the input length $k = (N+1)$ and the depth $d = O(\log N)$ of the circuits $\{C_r\}$ defined above.
- $\mathsf{OT.Query}(\mathsf{OT.crs}, b\,;\, r \in \{0,1\}^N)$: Compute $\hat{b} = b \oplus \mathsf{Ext}(r;\mathsf{seed})$, and compute $\mathsf{digest} = \mathsf{Compress}(\mathsf{LFE.crs}, C_r)$. Output:
$$M^{(1)} = \left(\hat{b}, \mathsf{digest}\right).$$
- $\mathsf{OT.Response}(\mathsf{OT.crs}, M^{(1)}, \mu_0, \mu_1)$: Define attributes $x_0 = (\mathsf{seed}, \hat{b})$ and $x_1 = (\mathsf{seed}, 1 \oplus \hat{b})$. Output
$$M^{(2)} = (\mathsf{ct}_0, \mathsf{ct}_1) \quad : \quad \mathsf{ct}_0 \leftarrow \mathsf{Enc}\left(\mathsf{LFE.crs}, \mathsf{digest}, (x_0, \mu_0)\right), \mathsf{ct}_1 \leftarrow \mathsf{Enc}\left(\mathsf{LFE.crs}, \mathsf{digest}, (x_1, \mu_1)\right).$$
- $\mathsf{OT.Output}(M^{(2)}, b, r)$: Output:
$$\mu = \mathsf{Dec}(\mathsf{LFE.crs}, C_r, \mathsf{ct}_b).$$

Claim 5.5 (Correctness)**.** *The OT is correct.*

Proof. If the receiver bit is b, we have $C_r(x_b) = b \oplus \hat{b} \oplus \mathsf{Ext}(r;\mathsf{seed}) = b \oplus (b \oplus \mathsf{Ext}(r;\mathsf{seed})) \oplus \mathsf{Ext}(r;\mathsf{seed}) = 0$. Therefore, by the correctness of the AB-LFE the ciphertext ct_b decrypts to μ_b. □

Claim 5.6 (Sender Security)**.** *The OT satisfies Sender Security (with Semi-Malicious Receiver).*

Proof. This follows by selective-security of the underlying AB-LFE.

Let EXP^ρ_{OT} be the OT Sender Security game. We define a hybrid game $\mathbf{H}$ which is the same as EXP^ρ_{OT} except that we compute the response $M^{(2)}$ by choosing $\mathsf{ct}_b \leftarrow \mathsf{Enc}\left(\mathsf{LFE.crs}, \mathsf{digest}, (x_b, \mu_b)\right), \mathsf{ct}_{1-b} \leftarrow \mathsf{LFE.Sim}(\mathsf{LFE.crs}, \mathsf{digest}, C_r, x_{1-b})$. We show that for $\rho \in \{0,1\}$ we have $\text{EXP}^\rho_{OT}(1^\lambda)$ is indistinguishable from $\mathbf{H}$ which proves the claim.

Let Adv be an adversary that distinguishes EXP^ρ_{OT} from $\mathbf{H}$. We show how to use Adv to break the selective security of AB-LFE. First chooses $(\beta, \mu) \leftarrow \{0,1\}^2$ uniformly at random. Choose $\mathsf{seed} \leftarrow \{0,1\}^N$ and give the challenger the AB-LFE input (x, μ) consisting of the attribute $x = (\beta, \mathsf{seed})$ and message μ. The challenger replies with $\mathsf{LFE.crs}$. Send $\mathsf{OT.crs} = (\mathsf{LFE.crs}, \mathsf{seed})$ to Adv who responds with the selection (r, b, μ_0, μ_1). If $\beta = \mathsf{Ext}(r;\mathsf{seed})$ and $\mu = \mu_{\rho \oplus (1-b)}$ then continue else terminate and output a random bit. Let $\hat{b} = \mathsf{Ext}(r;\mathsf{seed}) \oplus b$, $x_0 = (\mathsf{seed}, \hat{b})$ and $x_1 = (\mathsf{seed}, 1 \oplus \hat{b})$. This implies $x_{1-b} = (\mathsf{seed}, \rho)$. Give the circuit C_r to the challenger and get back a ciphertext ct_{1-b}. Compute $\mathsf{ct}_b \leftarrow \mathsf{Enc}\left(\mathsf{LFE.crs}, \mathsf{digest}, (x_b, \mu_b)\right)$ and give $M^{(2)} = (\mathsf{ct}_0, \mathsf{ct}_1)$ to Adv. Output whatever Adv outputs.

Note that the probability of the above reduction terminating is $1/4$ and is independent of the choices of Adv or the AB-LFE challenger. Conditioned on not terminating, if the AB-LFE challenger is computing

a real ciphertext then the above corresponds to EXP^{ρ}_{OT}, where the ct_{1-b} is a real encryption of $\mu_{\rho\oplus(1-b)}$ with attribute x_{1-b}, and if the challenger is using a simulated ciphertext then the above corresponds to $\mathbf{H}$, where ct_{1-b} is a simulated ciphertext. Therefore if Adv can distinguish these with advantage ε then the above reduction has advantage $\varepsilon/4$ in the selective AB-LFE game.

Notice that the proof covers the semi-malicious case, where security holds *for all* r chosen adversarially (that is, even when r is not necessarily chosen randomly). This is because the AB-LFE experiment allows its adversary to pick the challenge circuit C_r arbitrarily.

□

Claim 5.7 (Statistical Receiver Security)**.** *The OT achieves statistical Receiver Security.*

Proof. We argue here that the choice bit b is statistically hidden given

$$\mathsf{OT.crs} = (\mathsf{LFE.crs}, \mathsf{seed}), M^{(1)} = \left(\hat{b}, \mathsf{digest}\right)$$

where $\hat{b} = b \oplus \mathsf{Ext}(r; \mathsf{seed})$, and $\mathsf{digest} = \mathsf{Compress}(\mathsf{LFE.crs}, C_r)$.

By Lemma 2.5 and the fact that the AB-LFE is "sufficiently compressing" we have $\mathbf{H}_\infty(r|\mathsf{digest}) \geq N - |\mathsf{digest}| \geq \omega(\log\lambda)$. Therefore the Leftover Hash Lemma ensures that $\mathsf{Ext}(r; \mathsf{seed})$ is statistically close to uniform even given $\mathsf{OT.crs}, \mathsf{digest}$. Therefore:

$$(\mathsf{OT.crs}, (\hat{0} = \mathsf{Ext}(r; \mathsf{seed}), \mathsf{digest})) \overset{s}{\approx} (\mathsf{OT.crs}, (\mathcal{U}_1, \mathsf{digest})) \overset{s}{\approx} (\mathsf{OT.crs}, (\hat{1} = 1 \oplus \mathsf{Ext}(r; \mathsf{seed}), \mathsf{digest}))$$

where $\mathcal{U}_1$ is a uniformly random bit. This proves the claim.

□

6 Applications

6.1 "Bob-Optimized" 2-Round 2PC

We present a construction of a 2-round 2-party computation (2PC) protocol for a function $f : \{0,1\}^{k_A} \times \{0,1\}^{k_B} \to \{0,1\}^{\ell}$ where Alice has input $x_A \in \{0,1\}^{k_A}$, Bob has input $x_B \in \{0,1\}^{k_B}$ and Alice learns $y = f(x_A, x_B)$ while Bob does not learn anything. Without loss of generality, Alice initiates the protocol by sending the first round message to Bob and learns her output after receiving the second round message from Bob. This is the first such protocol with the following properties:

- The total communication complexity is smaller than Alice's input size k_A.
- Bob's computational complexity is is smaller than the circuit-size of f or even the size of Alice's input k_A.

Let $\mathsf{LFE} = (\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ be a function-hiding LFE scheme as defined in Definition 5.1. We define the following protocol π_f, which is a 2-round protocol in the common random string (CRS) model.

CRS Generation: Sample the common random string $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, f.\mathsf{params})$.

First Round (Alice → Bob): Let $C = f(x_A, \cdot)$ be a circuit with Alice's input hard-coded. Alice computes $\mathsf{digest}_C \leftarrow \mathsf{Compress}(\mathsf{crs}, C; r)$ using random coins r. She sends digest_C to Bob.

Second Round (Bob → Alice): Bob computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, x_B)$ and sends ct to Alice.

Output: Alice computes $y = \mathsf{Dec}(\mathsf{crs}, C, r, \mathsf{ct})$.

We note that in the setting of "semi-honest" security, we can remove the CRS and simply have Alice sample it and include it together with her first-round message. In the CRS model, we show that the construction is even "semi-malicious" secure, meaning that the parties follows the protocol as specified but can use arbitrarily bad randomness. See [AJL+12] for a detailed definition. We consider a static corruption model where one of the parties can be corrupted before the protocol starts. We also distinguish between "selective-honest-input" and "adaptive-honest-input" security. In the former, the environment must choose the input of the honest party selectively before the CRS is chosen while in the latter (which is the standard notion in MPC) it can do so adaptively depending on the CRS. The input of the semi-maliciously corrupted party can always be chosen adaptively after the CRS is chosen.

Theorem 6.1. *Given a function-hiding LFE scheme, the protocol π_f is a semi-maliciously secure 2-round 2PC protocol with static corruptions in the common random string (CRS) model. If the function-hiding property holds statistically then the protocol has statistical security against a corrupted Bob. If the LFE is selectively secure, then the protocol is selective-honest-input secure and if the LFE is adaptively secure, then the protocol is adaptive-honest-input secure.*

Proof. The simulator always chooses $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, f.\mathsf{params})$ honestly. We consider two cases where either Alice or Bob is corrupted separately.

Firstly, we consider the case of a (semi-maliciously) corrupted Bob. Note that no matter what randomness Bob uses in the second round to compute the ciphertext ct, by perfect correctness we ensure that Alice gets the correct output $y = f(x_A, x_B)$. Therefore, the simulator's only goal is to simulate the the first message digest_C from Alice. It does so by setting C to be the circuit $C = f(0^{k_A}, \cdot)$ with Alice's input replaced by 0^{k_A} and sampling $\mathsf{digest}_C \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$. By the (statistical) function-hiding security of the LFE the above is (statistically) indistinguishable from the way digest_C is computed by Alice in the real world.

Secondly, in the case of a (semi-maliciously) corrupted Alice, the 2PC simulator gets the output $f(x_A, x_B)$ and has to simulate the ciphertext ct that Alice receives in the second round. We simply use the LFE simulator $\mathsf{LFE.Sim}(\mathsf{crs}, \mathsf{digest}_C, C, f(x_A, x_B))$ to do this and security follows directly from that of the LFE scheme. We rely on the fact that in the (function-hiding) LFE security definition, the digest digest_C can be chosen using adversarial randomness. If the LFE is only selectively secure, we need to ensure that Bob's input x_B is chosen by the environment before the CRS is chosen and therefore get a selective honest-input secure 2PC. □

Instantiation and Parameters. Under the LWE assumption, the above protocol π_f can be instantiated to get selective-honest-input semi-malicious security in the CRS model or semi-honest security in the plain model. For a circuit $f : \{0,1\}^{k_A} \times \{0,1\}^{k_B} \to \{0,1\}^\ell$ with depth d we get the following parameters:

- The size of the CRS is $k_B \cdot \mathsf{poly}(\lambda, d)$.
- The communication complexity is $\widetilde{O}(k_B + \ell) \cdot \mathsf{poly}(\lambda, d)$. Alice's first-round message is of size $\mathsf{poly}(\lambda)$ and Bob's second-round message is of size $\widetilde{O}(k_B + \ell) \cdot \mathsf{poly}(\lambda, d)$.
- Alice's computation complexity is $\widetilde{O}(|f|) \cdot \mathsf{poly}(\lambda, d)$. Bob's computation complexity is $\widetilde{O}(k_B + \ell) \cdot \mathsf{poly}(\lambda, d)$.

Under the adaptive LWE (ALWE) assumption with optimistic parameters (see Section 4.3), we get adaptive-honest-inputs (the usual notion) semi-malicious security with and the same parameters as above. Under the ALWE assumption with provable parameters, which follows from the sub-exponential security of LWE, we get adaptive-honest-inputs semi-malicious security where efficiency degrades as follows:

- The size of the CRS is $\mathsf{poly}(\lambda)$.

- The communication complexity is $\widetilde{O}(\ell) \cdot \mathsf{poly}(\lambda, k_B, d)$.
- Alice's computation complexity is $\widetilde{O}(|f|) \cdot \mathsf{poly}(\lambda, k_B, d)$. Bob's computation complexity is $\widetilde{O}(\ell) \cdot \mathsf{poly}(\lambda, k_B, d)$.

We can rely on the result of [AJL+12] which shows that we can compile any semi-malicious protocol into one with fully malicious security by additionally relying on non-interactive zero-knowledge proofs of knowledge (NIZK-PoK) in the CRS model. This preserves round-complexity but unfortunately does not preserve communication and computation complexity. In particular, Alice needs to prove a complex statement that she computed digest_C correctly, meaning that she knows some x_A, r such that $\mathsf{digest}_C = \mathsf{Compress}(\mathsf{crs}, f(x_A, \cdot); r)$, in a way that Bob can verify much more efficiently than running f. To do so, we need to rely on *succinct* NIZK-PoKs, also known as ZK-SNARKs [Mic94, Gro10, BCCT13, BCI+13, GGPR13], which allow us to preserve the communication and computation complexity. Such ZK-SNARKs exist in the random-oracle model or under various "knowledge/extractability" assumptions. Using ZK-SNARKs, the above-stated results and parameters hold with malicious security.

Observations. If the first message of a 2-round 2PC is shorter than k_A, then it can be considered as a digest of an LFE, which implies that a CRS is required, as shown in Section B. We also recall that the result of [HW15] shows that any semi-malicious secure 2PC has to have communication complexity which exceeds the output size ℓ.[9]

6.2 MPC without Online Computation

We now construct an MPC protocol where N parties with respective inputs $x_1, \ldots, x_N$ can securely evaluate some function $y = f(x_1, \ldots, x_N)$ over their inputs so that every party learns y. The protocol consists of three phases. There is a *pre-processing phase* in which each party does some local deterministic computation over the circuit f, but independent of the party's input. This phase can be reused across many protocol executions. Then there is an *online* phase in which the parties communicate with each other and execute some protocol. Lastly, there is a *post-processing phase* in which each party does some local computation over the protocol transcript to recover the output y.

Let $\mathsf{LFE} = (\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ be an LFE scheme as in Definition 3.1 (with a deterministic compression function and without function hiding). Let ψ be a generic MPC protocol without any special efficiency requirements. For a function $f : (\{0,1\}^k)^N \to \{0,1\}^\ell$, we define the following two related protocols π_f^{SH} (a semi-honest secure protocol in the plain model) and π_f^M (a malicious secure protocol in the CRS model):

The Protocol π_f^M. A malicious secure protocol in the CRS model.

- *CRS Generation:* Sample $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\lambda, f.\mathsf{params})$.
- *Pre-Processing:* Each party computes $\mathsf{digest}_f = \mathsf{LFE.Compress}(\mathsf{crs}, f)$.
- *Online Phase:* The parties run a generic MPC protocol ψ for the function

$$\mathsf{ct} = \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_f, (x_1, \ldots, x_N)\,; \bigoplus_{i \in [N]} r_i).$$

  We think of $\mathsf{crs}, \mathsf{digest}_f$ as part of the public function description. Each party i has input (x_i, r_i), where r_i is chosen uniformly at random, and gets ct as output.
- *Post-Processing:* Each party computes $y = \mathsf{LFE.Dec}(\mathsf{crs}, f, \mathsf{ct})$.

[9] This result does not hold in the CRS model in general, but it does hold in the non-programmable CRS model where the simulator gets the CRS as input but cannot choose it, which is the case for us.

The Protocol π_f^{SH}. A a semi-honest secure protocol in the plain model.

- *Pre-Processing:* Party 1 samples $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\lambda, f.\mathsf{params})$ and computes $\mathsf{digest}_f = \mathsf{LFE.Compress}(\mathsf{crs}, f)$.
- *Online Phase:* The parties run a generic MPC protocol ψ for the function

$$\mathsf{ct} = \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_f, (x_1, \ldots, x_N)\,;\, \bigoplus_{i\in[N]} r_i).$$

  The input of party 1 consists of $(\mathsf{crs}, \mathsf{digest}_f, x_1, r_1)$ and the input of each party $i > 1$ consists of (x_i, r_i). The values r_i are chosen uniformly at random. Each party get $\mathsf{crs}, \mathsf{ct}$ as output.

- *Post-Processing:* Each party computes $y = \mathsf{LFE.Dec}(\mathsf{crs}, f, \mathsf{ct})$.

Theorem 6.2. *Assuming that the LFE scheme is selectively secure and that ψ is a semi-honest secure MPC, the protocol π_f^{SH} is a semi-honest secure MPC for f in the plain model. Assuming that the LFE is adaptively secure and that ψ is a malicious secure MPC, the protocol π_f^{M} is a malicious secure MPC for f in the CRS model. In both cases we assume static corruptions.*

Proof. We begin by replacing the invocation of the protocol ψ with an ideal functionality $\mathcal{F}_{\mathsf{Enc}}$ that computes $\mathsf{LFE.Enc}$. We can do so by security of ψ in the appropriate setting.

Then we show indistinguishability between the ideal world with a functionality $\mathcal{F}_f$ that computes f and the $\mathcal{F}_{\mathsf{Enc}}$-hybrid world in which the parties run the protocol π_f with invocations of ψ replaced by $\mathcal{F}_{\mathsf{Enc}}$. Our simulator in the ideal world does the following:

- It runs $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\lambda, f.\mathsf{params})$ and gives crs to all parties.
- It gets the inputs x_i that the corrupted parties send to the ideal functionality $\mathcal{F}_{\mathsf{Enc}}$ in the online phase. It sends these inputs to the functionality $\mathcal{F}_f$ and gets an output y.
- It runs $\mathsf{ct} \leftarrow \mathsf{LFE.Sim}(\mathsf{crs}, \mathsf{digest}_f, f, y)$ and sets it as the output of $\mathcal{F}_{\mathsf{Enc}}$.

The only difference between the simulation in the ideal world and the $\mathcal{F}_{\mathsf{Enc}}$-hybrid world is how ct is generated: in the real world it is generated via $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_f, (x_1, \ldots, x_N)\,;\, \bigoplus_i r_i)$ and since there is at least one uncorrupted party we can write this as $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_f, (x_1, \ldots, x_N)\,;\, r^*)$ where r^* is uniformly random and unknown to the adversary. In the ideal world ct is generated via $\mathsf{ct} \leftarrow \mathsf{LFE.Sim}(\mathsf{crs}, \mathsf{digest}_f, f, y)$ where $y = f(x_1, \ldots, x_N)$. These are indistinguishable by the security of the LFE. Note that in the malicious setting some of the x_i values are chosen adaptively depending on the CRS and therefore we need adaptive LFE security. □

Instantiation and Parameters. We can instantiate the protocol ψ with (e.g.,) the [GMW86] protocol, which provides malicious security, using oblivious transfer based on LWE. By plugging in the parameters for LFE based on LWE we get the following parameters. For a circuit $f : (\{0,1\}^k)^N \to \{0,1\}^\ell$ with depth d we get a semi-hones MPC protocol in the plain model and a malicious-secure MPC in the CRS model with the following efficiency:

- In the CRS model, the size of the CRS is $k \cdot N \cdot \mathsf{poly}(\lambda, d)$.
- The communication complexity is $(k + \ell) \cdot \mathsf{poly}(\lambda, d, N)$.
- Each party's computation in the online phase is $(k + \ell) \cdot \mathsf{poly}(\lambda, d, N)$.
- Each party's computation in the pre-processing/post-processing phase is $|f| \cdot \mathsf{poly}(\lambda, d)$.

Semi-honest security holds under the LWE assumption. Malicious security in the CRS model holds under the adaptive LWE (ALWE) assumption with optimistic parameters. Under the ALWE assumption with provable parameters, which follows from the sub-exponential security of LWE, we get a malicious secure protocol in the CRS model with decreased efficiency where all of the $\mathsf{poly}(\lambda, d)$ and $\mathsf{poly}(\lambda, d, N)$ factors become $\mathsf{poly}(\lambda, d, k, N)$.

Improving 2-round 2PC. Alternatively, we can instantiate the protocol ψ with a 2-round semi-honest protocol in the plain model such as the one in [BL18, GS18] based only on OT which follows from LWE. In that case, we get semi-honest MPC with the same efficiency as above but with only 2 rounds of interaction in the plain model. Note that the total computational complexity of each party, including pre-processing and post-processing, is $|f| \cdot \mathsf{poly}(\lambda, d) + (k + \ell) \cdot \mathsf{poly}(\lambda, d, N)$. Despite many works on 2-round 2-PC [MW16, PS16, BP16, GS17, GS18, BL18], in all prior constructions the per-party computation is at least $|f| \cdot N^2$ even in the semi-honest setting and even in the CRS model. Therefore the above gives an asymptotic improvement on the computation complexity of 2-round 2-PC even if we do not distinguish between online and offline computation.

Necessity of Pre-Processing and Post-Processing. We argue that both the pre-processing and post-processing steps are necessary, at least in the circuit model where the function f is given to the parties as a circuit.

Assume that we were able to remove the pre-processing step. Since the online computation is small, most bits of the description of f were never accessed by any party during the protocol until the post-processing step. But that means that, during the post-processing step, party 1 can modify a random bit of f (in its head) to get f' and with good probability is able to learn $f'(x_1, \ldots, x_N)$. This violates security. (Note that this argument relies on f having a large description. It remains an interesting problem if we can get rid of pre-processing if f is a Turing Machine with short description.)

Assume we were able to remove the post-processing step. This means that, even if we didn't care about security, we now have a method to pre-process an arbitrary circuit f and then evaluate $f(x)$ on arbitrary inputs x at a lower cost than computing f. But that means that there is an altogether smaller circuit g that computes f. Since not all circuits are compressible [Sha49] this cannot be done in general.

Acknowledgements. We thank Yuval Ishai for insightful discussions on succinct secure computation.

References

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, Heidelberg, March / April 2015.

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 3–35. Springer, Heidelberg, August 2017.

[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 David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501. Springer, Heidelberg, April 2012.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 333–362. Springer, Heidelberg, August 2016.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 111–120. ACM Press, June 2013.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 315–333. Springer, Heidelberg, March 2013.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, Heidelberg, May 2014.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12: 19th Conference on Computer and Communications Security*, pages 784–796. ACM Press, October 2012.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round MPC from k-round OT via garbled interactive circuits. In *EUROCRYPT*, 2018.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumption. In *EUROCRYPT*, 2018.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 190–213. Springer, Heidelberg, August 2016.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, Heidelberg, March 2011.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 264–302. Springer, Heidelberg, November 2017.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society Press, October 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014: 5th Innovations in Theoretical Computer Science*, pages 1–12. Association for Computing Machinery, January 2014.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, Heidelberg, February 2007.

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

Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65. Springer, Heidelberg, August 2017.

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569. Springer, Heidelberg, August 2017.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, March 2008.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, Heidelberg, May 2013.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564. ACM Press, June 2013.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187. IEEE Computer Society Press, October 1986.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, Heidelberg, December 2010.

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In *58th Annual Symposium on Foundations of Computer Science*, pages 588–599. IEEE Computer Society Press, 2017.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT*, 2018.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, Heidelberg, August 2013.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, Heidelberg, August 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 545–554. ACM Press, June 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523. Springer, Heidelberg, August 2015.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Innovations in Theoretical Computer Science*, pages 163–172. Association for Computing Machinery, January 2015.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24. ACM Press, May 1989.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, Heidelberg, April 2008.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science*, pages 436–453. IEEE Computer Society Press, November 1994.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, Heidelberg, April 2012.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763. Springer, Heidelberg, May 2016.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342. ACM Press, May / June 2009.

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 217–238. Springer, Heidelberg, October / November 2016.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 463–472. ACM Press, October 2010.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, Heidelberg, May 2005.

[Wee17] Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 206–233. Springer, Heidelberg, November 2017.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society Press, November 1982.

A Direct Construction of Function Hiding LFE

We show a natural and efficient way to upgrade the constructions from Section 4.1 and 4.2 to achieve *statistical* Function Hiding. Then, given that the transformation of Section 4.4 preserves (statistical) Function Hiding (as the digest in the transformation is a digest from the underlying LFE), this gives an alternative construction to get a statistical Function Hiding LFE from LWE.

The idea is to make the digest digest_C output by the Compression algorithm $\mathsf{Compress}$ (statistically close to) uniform by adding a random subset sum of extra random matrices. On a high level, this corresponds to adding to the output an inner product (over the integers) of additional input bits with a random binary vector. Then, Function Hiding is guaranteed by the Leftover Hash Lemma, while correctness is preserved by considering that the Encryption algorithm implicitely sets those new inputs to zero (so that the added inner product equals 0).

For simplicity, we present the construction for the 1-bit AB-LFE scheme (defined in Section 4.1). This can be directly combined with the modification in Section 4.2 to get Statistical Function Hiding with many output bits (with an *additive* overhead in the ciphertext), and also imply a Statistical Function Hiding for LFE via the compiler of Section 4.4.

Set parameters $n, m, q, B, \sigma, \sigma_F$ as in Section 4.1, and $N = 2nm\lceil \log q \rceil$.

- $\mathsf{crsGen}(1^\lambda)$: Output $n + N$ random matrices in $\mathbb{Z}_q^{n\times m}$:

$$\mathsf{crs} = (\{\mathbf{A}_i\}_{i\le k}\,,\, \{\mathbf{B}_i\}_{i\le N})\,.$$

- $\mathsf{Compress}(\mathsf{crs}, C)$: Compute $\mathbf{A}_C = \mathsf{EvalPK}(C, \{\mathbf{A}_i\}_{i\le k})$. Pick $\mathbf{r} \leftarrow \{0,1\}^N$ and output:

$$\mathsf{digest}_C = \widetilde{\mathbf{A}}_C = \mathbf{A}_C + \sum_{i=1}^{N} r_i \mathbf{B}_i.$$

- $\mathsf{Enc}(\mathsf{crs}, \tilde{\mathbf{A}}_C, (x, \mu))$: Pick $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_i \leftarrow \chi^m$ for $i \le k$, and compute:

$$\mathbf{b}_i = \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G}) + \mathbf{e}_i^\top, \quad i \le k.$$

  Sample $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\$)$, set $\mathbf{d} = \widetilde{\mathbf{A}}_C \cdot \mathbf{t}$, and compute

$$\beta = \mathbf{s}^\top \mathbf{d} + \widetilde{e} + \mu \cdot \lfloor q/2 \rceil,$$

  where $\widetilde{e} \leftarrow [-B', B']$, and additionally compute $\gamma_{\mathbf{i}} = \mathbf{s}^\top \mathbf{b}_i + \mathbf{e}'^\top_{\mathbf{i}}$ for $i \le N$, where $\mathbf{e}'_{\mathbf{i}} \leftarrow \chi^m$. Output:

$$\mathsf{ct} = (\{\mathbf{b}_i\}_{i\le k},\, \{\gamma_{\mathbf{i}}\}_{i\le N},\, \beta,\, \mathbf{t},\, x).$$

- $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct}, \mathbf{r})$: If $C(x) = 1$ output $\perp$. Else compute:

$$\widetilde{\mathbf{b}} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i\le k}, \{\mathbf{b}_i\}_{i\le k}, x) + \sum_{i=1}^{N} r_i \gamma_{\mathbf{i}},$$

  and output $\lfloor (\beta - \widetilde{\mathbf{b}}^\top \mathbf{t})/q \rceil$.

$$\widetilde{\mathbf{b}} = \mathsf{EvalCT}(C, \{\mathbf{A}_i\}_{i\le k}, \{\mathbf{b}_i\}_{i\le k}, x)\,,$$

  where EvalCT is defined in Section 2.3. Output $\lfloor (\beta - \widetilde{\mathbf{b}}^\top \mathbf{t})/q \rceil$.

Correctness. Intuitively, adding $\sum_{i=1}^{N} r_i\gamma_i$ to $\widetilde{\mathbf{b}}$ corresponds to adding $\langle \mathbf{r}, \widetilde{\mathbf{x}} \rangle$ (over the integers) to the initial message, where $\widetilde{\mathbf{x}}$ corresponds to N extra input bits (that would be encoded along using matrices $\{\mathbf{B}_i\}_{i\leq N}$). The Encryption algorithm, by setting $\gamma_{\mathbf{i}} = \mathbf{s}^\top \mathbf{b}_i + \mathbf{e}'^\top_{\mathbf{i}}$, implicitly sets $\widetilde{\mathbf{x}} = \mathbf{0}$, so that correctness is maintained.

More formally, we have by correctness of EvalCT: $\widetilde{\mathbf{b}} = \mathbf{s}^\top[\mathbf{A}_C - C(x)\mathbf{G}] + \mathbf{e}_C + \mathbf{s}^\top(\sum_{i\leq N} r_i\mathbf{b}_i) + \left(\sum_{i\leq N} r_i\mathbf{e}_i^\top\right)$. In particular, if $C(x) = 0$ then

$$|\widetilde{\mathbf{b}} - \mathbf{s}^\top\mathbf{d}| \leq (m+1)^{d+1} \cdot B + B' + N \cdot B < q/4.$$

AB-LFE security. AB-LFE security is preserved in this new construction: the simulator additionally outputs uniform vectors $\{\gamma_{\mathbf{i}}\}_{i\leq N}$ as a part of the ciphertext. In particular, we also switch $\{\gamma_{\mathbf{i}}\}$ to uniform vectors when arguing LWE (between Hybrid 2 and Hybrid 3, where all the hybrids are defined very similarly as in Section 4.1)).

Function Hiding. Note that if q is a prime, and N and K are integers, the function

$$H_{\mathbf{M}} : \mathbf{r} \in \{0,1\}^N \mapsto \mathbf{r}^\top\mathbf{M},$$

where $\mathbf{M} \in \mathbb{Z}_q^{N\times K}$ is a 2-universal hash function.

Therefore, the Leftover Hash Lemma states that given random matrices $\mathbf{B}_i$, we have that $\sum_{i=1}^{N} r_i\mathbf{B}_i$ (and therefore $\mathsf{digest}_C \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$) is statistically close to uniform over the randomness of $\mathbf{r} \leftarrow \{0,1\}^N$ by setting:

$$\mathbf{M} = \begin{bmatrix} \mathbf{B}_1[1,1] & \cdots & \mathbf{B}_1[n,m] \\ \vdots & \ddots & \vdots \\ \mathbf{B}_N[1,1] & \cdots & \mathbf{B}_N[n,m] \end{bmatrix} \in \mathbb{Z}_q^{N\times K},$$

and setting $K = nm\lceil \log q \rceil$.

B Necessity of a CRS

In this section we show that LFE and even AB-LFE requires a CRS, at least if we consider non-uniform adversaries. This is for the same reason that collision resistant hash functions need a CRS. In fact, we show that the AB-LFE compression function is a collision-resistant hash function.

AB-LFE Compression is Collision Resistant. Consider the class of functions $f_y(x) := \langle y, x\rangle$, where $x, y \in \{0,1\}^k$ for some k. Assume we have a (selectively secure) AB-LFE scheme which is "sufficiently compressing", meaning that for a sufficiently large $k = k(\lambda) = \mathsf{poly}(\lambda)$, if $\mathsf{digest} = \mathsf{Compress}(\mathsf{crs}, f_y)$ then $|\mathsf{digest}| < k$. Then $H_{\mathsf{crs}}(y) = \mathsf{Compress}(\mathsf{crs}, f_y)$ is a Collision-Resistant Hash Function with the seed crs.

Claim B.1 ((AB-)LFE is Collision Resistant)**.** *Suppose* $(\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ *is a selectively secure AB-LFE which is "sufficiently compressing". Then* $H_{\mathsf{crs}}(y) = \mathsf{Compress}(\mathsf{crs}, f_y)$ *is a Collision-Resistant Hash Function.*

Proof. Suppose that an adversary $\mathcal{A}$ breaks the collision resistance on H. We build a reduction $\mathcal{R}$ that breaks the security of the AB-LFE as follows.

$\mathcal{R}$ declares a random challenge attribute x^* and a random message μ. It then receives a CRS, and forwards it to $\mathcal{A}$, who outputs a collision $y \neq y'$ for H_{crs} with some non-negligible advantage.

$\mathcal{R}$ receives a challenge ciphertext ct. Note that as $y \neq y'$ and $f.(\cdot)$ 2-universal, we have that $f_y(x^*) \neq f_{y'}(x^*)$ with probability $1/2$ (over the randomness of x^* alone); the reduction aborts if it is not the case. If it doesn't, suppose without loss of generality that $f_y(x^*) = 0$ and $f_{y'}(x^*) = 1$.

The reduction then chooses C which computes $f_{y'}(\cdot)$, so that $f_{y'}(x^*) = 1$ in the LFE experiment. It then receives a challenge ciphertext ct.

Now, by correctness of the AB-LFE, the reduction can, in the real experiment, decrypt the ciphertext using f_y and recover μ; but in the Ideal one the ciphertext is independent of μ, which allows to distinguish the real and the ideal experiments with non-negligible probability.

□

Necessity of a CRS. This construction shows the necessity of using a CRS in the compression algorithm for AB-LFE. Indeed, without a CRS, a non-uniform adversary can store such a collision $y \neq y'$ as advice, and use it to break security.

"Very Selective" AB-LFE imply UOWHFs. We can also consider an alternative definition of LFE security, where the adversary has to declare the circuit C and the input x *before* seeing the CRS, which we call "Very Selective" security.

The reduction above can be directly extended to show that the compression algorithm of a "Very Selective" LFE applied on a 2-universal hash function is a Universal One-Way Hash Function (UOWHF). The only difference is that the adversary for the UOWHF chooses y and sends it to the reduction, who sets $f_y(\cdot)$ as a challenge circuit *before* seeing the CRS.

The same argument as above shows that a CRS is also necessary in that case.

C LFE for Simple Functions

In this section, we present LFE protocols for the class of linear functions under the k-Linear assumption in prime-order cyclic groups. Here, we reserve k for k-Linear, and n for the length of the inputs to the circuit.

C.1 LFE for Linear Functions from k-Lin

We present a LFE for the class of linear functions $C_{\mathbf{y}} : \mathbf{x} \mapsto \mathbf{x}^\top \mathbf{y}$. Functionality only works when the output is small. The construction is very similar to the functional encryption scheme for linear functions in [ABDP15, ALS16, Wee17].

Construction. Fix a prime order group $\mathbb{G}$ of order q, and we will use the implicit representation notation for group elements where $[\cdot]$ denote entry-wise exponentiation for a matrix. We will rely on the k-Linear (and more generally MDDH assumption) which says that

$$[\mathbf{A}], [\mathbf{A}\mathbf{s}] \approx_c [\mathbf{A}], [\mathbf{c}]$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{(k+1)\times k}, \mathbf{s} \leftarrow \mathbb{Z}_q^k, \mathbf{c} \leftarrow \mathbb{Z}_q^{k+1}$.

- $\mathsf{crsGen}(1^\lambda)$: Pick $\mathbf{A} \leftarrow \mathbb{Z}_q^{k\times n}$ and outputs
$$\mathsf{crs} = [\mathbf{A}] \in \mathbb{G}^{k\times n}.$$
- $\mathsf{Compress}(\mathsf{crs}, C_{\mathbf{y}})$: Output:
$$\mathsf{digest}_C = [\mathbf{A}\mathbf{y}] \in \mathbb{G}^{k\times 1}.$$
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, \mathbf{x})$: Pick $\mathbf{s} \leftarrow \mathbb{Z}_q^k$, and compute:
$$\mathsf{ct} = ([\mathbf{b}] = [\mathbf{s}^\top \mathbf{A} + \mathbf{x}^\top], [\beta] = [\mathbf{s}^\top \mathbf{A}\mathbf{y}]) \in \mathbb{G}^{1\times(n+1)}.$$
- $\mathsf{Dec}(\mathsf{crs}, \mathbf{y}, \mathsf{ct})$: Output the discrete log of $[\mathbf{b}\mathbf{y} - \beta]$.

Correctness. Correctness is straightforward.

Adaptive security. We first write down the simulator:

- simulate the crs and digest_C honestly as in crsGen and $\mathsf{Compress}$.
- on input $a = C_{\mathbf{y}}(\mathbf{x})$ from the ideal functionality, pick $\mathbf{b} \leftarrow \mathbb{Z}_q^{1\times n}$ and output

$$\mathsf{ct} = ([\mathbf{b}], [\mathbf{by} - a]).$$

To prove that the simulator works, we consider an intermediate hybrid as before, where $[\mathbf{b}]$ is computed as in the honest Enc, and β is computed (given $[\mathbf{b}], C_{\mathbf{y}}(\mathbf{x})$) as in the simulator. This distribution is identically distributed to the real distribution, and computationally indistinguishable from the simulated one via the k-Linear assumption applied to $[\mathbf{A}], [\mathbf{s}^\top \mathbf{A}]$.

C.2 LFE for Linear Predicates from k-Lin

Next, we consider the "predicate" variant where

$$C_{\mathbf{y}}(\mathbf{x}, \mu) = \begin{cases} \mu & \text{if } \mathbf{x}^\top \mathbf{y} = 0 \\ \bot & \text{otherwise} \end{cases}$$

This basically follows from applying the previous construction to the vectors $v\mathbf{x} \mid \mu$ where $v \leftarrow \mathbb{Z}_q$ is chosen by Enc and $\mathbf{y} \mid 1$.

- $\mathsf{crsGen}(1^\lambda)$: pick $\mathbf{A} \leftarrow \mathbb{Z}_q^{k\times n}$ and outputs

$$\mathsf{crs} = [\mathbf{A}] \in \mathbb{G}^{k\times n}$$

- $\mathsf{Compress}(\mathsf{crs}, C_{\mathbf{y}})$: Output:

$$\mathsf{digest}_C = [\mathbf{Ay}] \in \mathbb{G}^{k\times 1}$$

- $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_C, (\mathbf{x}, \mu))$: Pick $\mathbf{s} \leftarrow \mathbb{Z}_q^k, v \leftarrow \mathbb{Z}_q$, and compute:

$$\mathsf{ct} = ([\mathbf{b}] = [\mathbf{s}^\top \mathbf{A} + v\mathbf{x}^\top)], [\beta] = [\mathbf{s}^\top \mathbf{Ay}] \cdot \mu) \in \mathbb{G}^{1\times(n+1)}.$$

- $\mathsf{Dec}(\mathsf{crs}, \mathbf{y}, \mathsf{ct})$: Output $[\beta] \cdot ([\mathbf{by}])^{-1}$.

Correctness. Correctness is straightforward.

Adaptive security. We first write down the simulator:

- simulate the crs and digest_C honestly as in crsGen and $\mathsf{Compress}$.
- on input $a = C_{\mathbf{y}}(\mathbf{x}, \mu)$ from the ideal functionality (it is convenient to think of a as being either μ or random), pick $\mathbf{b} \leftarrow \mathbb{Z}_q^{1\times n}$ and output

$$\mathsf{ct} = ([\mathbf{b}], [\mathbf{by} - a]).$$

To prove that the simulator works, we consider an intermediate hybrid as before, where $[\mathbf{b}]$ is computed as in the honest Enc, and β is computed (given $[\mathbf{b}], C_{\mathbf{y}}(\mathbf{x})$) as in the simulator. This distribution is identically distributed to the real distribution, and computationally indistinguishable from the simulated one via the k-Linear assumption applied to $[\mathbf{A}], [\mathbf{s}^\top \mathbf{A}]$.

D LFE from LWE – A Direct Construction

We describe how to use the "dual use" technique from BTVW [BTVW17] (used to obtain simpler constructions of predicate encryption for circuits from LWE ([GVW15]) to obtain a direct construction of LFE from LWE. The idea is to use the same secret vector $\mathbf{s}$ for the BGG+ encodings ([BGG$^+$14]), and the FHE encryption from [GSW13] of the attribute $\mathbf{x}$.

Let $\Psi_1, \ldots, \Psi_k$ denote GSW encryptions of $\mathbf{x}$ under the key $\mathbf{s}$ and let Ψ_f denote an encryption of $f(\mathbf{x})$ obtained via homomorphic evaluation of C. Suppose for simplicity that FHE decryption is given by

$$\mathbf{s}^\top \Psi_f \approx C(\mathbf{x}) \cdot \mathbf{s}^\top \mathbf{G}$$

Now, we also give out BGG+ encodings of the binary representation of $\Psi_1, \ldots, \Psi_k$ under some public matrices $\mathbf{A}_i$'s under the same $\mathbf{s}$. As it turns out, given C, we can "recode" these encodings to something of the form:

$$\mathbf{s}^\top [\mathbf{A}_C + \Psi_f] \approx \mathbf{s}^\top [\mathbf{A}_C + C(\mathbf{x})\mathbf{G}]$$

which is exactly what we want!

The actual construction is a bit more involved because FHE decryption is actually given by

$$[\mathbf{s} \mid -1]^\top \Psi_f \approx C(\mathbf{x}) \cdot [\mathbf{s} \mid -1]^\top \mathbf{G}.$$

D.1 Construction

BTVW matrix computation. [BTVW17] extended the evaluation of matrices from Secetion 2.3 to deal with functions whose output is a matrix instead of a bit (we still treat the input as bits). Suppose $f : x_1, \ldots, x_\ell \mapsto \mathbf{X}_C$ where these matrices have the same dimensions as $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_\ell$. Then, we can recode

$$\mathbf{A}_1 - x_1\mathbf{G}, \ldots, \mathbf{A}_\ell - x_\ell\mathbf{G} \mapsto \mathbf{A}_C - \mathbf{X}_C$$

where $\mathbf{A}_C$ is deterministically derived from $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$. As before, we use EvalPK to denote the derivation of $\mathbf{A}_C$ from $\mathbf{A}_i$'s and EvalCT to denote the derivation of $\mathbf{s}^\top(\mathbf{A}_C - \mathbf{X}_C)$ from $\mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G})$'s. We defer the details to [BTVW17, Section 4.1] and the full version of this paper.

Notation. We use gadget matrices $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\log q}$ and we write $\overline{\mathbf{G}} \in \mathbb{Z}_q^{n\times(n+1)\log q}$ to denote all but the last row of $\mathbf{G}$. Given a circuit computing a function $f : \{0,1\}^\ell \to \{0,1\}$, and GSW FHE encryptions $\Psi := (\Psi_1, \ldots, \Psi_\ell)$ of $x_1, \ldots, x_\ell$, we write Ψ_C to denote $\mathsf{fhe.eval}(f, \Psi)$. Noting that Ψ_C is a matrix, we let $\underline{\Psi}_C$ denote the last row of Ψ_C, and $\overline{\Psi}_C$ to denote all but the last row of Ψ_C. In addition, we write $\hat{C}$ to denote the circuit that computes $\Psi \mapsto \overline{\Psi}_C$, namely it takes as input the bits of Ψ and outputs the matrix $\overline{\Psi}_C$.

LFE Protocol. Consider the class $\{C : \{0,1\}^k \to \{0,1\}$ has depth at most $d\}$.

- $\mathsf{crsGen}(1^\lambda)$: pick the common random string to be L random matrices:

  $$\mathsf{crs} = (\mathbf{A}_1, \ldots, \mathbf{A}_L) \in \mathbb{Z}_q^{n\times m}.$$

  where $L = k \cdot (n+1)^2 \cdot \log^2 q$.

- $\mathsf{Compress}(\mathsf{crs}, C)$: Output:

  $$\mathsf{digest}_C = \mathbf{A}_{\hat{C}} = \mathsf{EvalPK}(\hat{C}, \{\mathbf{A}_i\}_{i\le L}),$$

  where $\mathsf{EvalPK}, \hat{C}$ are defined as above.

- $\mathsf{Enc}(\mathsf{crs}, \mathbf{A}_C, \mathbf{x})$: Pick $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}'_i \leftarrow \chi^m$ for $i \le k$, and compute:

$$\Psi_i = \begin{pmatrix} \mathbf{B}_i \\ \mathbf{s}^\top \mathbf{B}_i + \mathbf{e}'_i \end{pmatrix} + x_i \mathbf{G}.$$

  Let $\hat{\mathbf{x}}$ denote the binary representation of $[\Psi_1 \mid \cdots \Psi_k]$. Compute

$$\mathbf{b}_i = \mathbf{s}^\top (\mathbf{A}_i - \hat{x}_i \overline{\mathbf{G}}) + \mathbf{e}_i^\top, \quad i \le L.$$

  Sample $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\$)$, and compute

$$\beta = \mathbf{s}^\top \cdot \mathbf{A}_{\hat{C}} \cdot \mathbf{t} + \widetilde{e},$$

  where $\widetilde{e} \leftarrow \widetilde{\chi}$. Output:

$$\mathsf{ct} = (\{\mathbf{b}_i\}_{i \le L}, \beta, \mathbf{t}, x).$$

- $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct})$: Compute Ψ_C along with

$$\widetilde{\mathbf{b}} = \mathsf{EvalCT}(\hat{C}, \{\mathbf{A}_i\}_{i \le L}, \{\mathbf{b}_i\}_{i \le L}, x),$$

  (where EvalCT is defined as above), and compute:

$$\eta = \beta - (\widetilde{\mathbf{b}} + \underline{\Psi}_C) \cdot \mathbf{t}.$$

  Output the most significant bit of η.

Correctness. Correctness follows from the fact that for honestly generated ciphertexts, $\widetilde{\mathbf{b}}$ satisfies

$$\widetilde{\mathbf{b}} \approx \mathbf{s}^\top [\mathbf{A}_{\hat{C}} - \overline{\Psi}_C]$$

In addition, we have

$$[\mathbf{s} \mid -1] \Psi_C \approx C(x) \cdot [\mathbf{s} \mid -1] \mathbf{G}$$

Putting the two together, we have

$$\beta - (\widetilde{\mathbf{b}} + \underline{\Psi}_C) \cdot \mathbf{t} \approx C(x) \cdot [\mathbf{s} \mid -1] \mathbf{G} \cdot \mathbf{t} \tag{3}$$

upon which correctness follows readily.

Security. The proof of security follows from the earlier strategy:

- First, simulate β in the challenge ciphertext using (3):

$$\beta \approx (\widetilde{\mathbf{b}} + \underline{\Psi}_C) \cdot \mathbf{t} + C(x) \cdot [\mathbf{s} \mid -1] \mathbf{G} \cdot \mathbf{t}$$

- As before, sample a uniformly random $\mathbf{u}$ and compute $\mathbf{t} \leftarrow \mathbf{G}^{-1}(\mathbf{u})$, and simulate β as

$$\beta \approx (\widetilde{\mathbf{b}} + \underline{\Psi}_C) \cdot \mathbf{t} + C(x) \cdot [\mathbf{s} \mid -1] \mathbf{u}$$

- Use LWE with secret $\mathbf{s}$ to replace the $\mathbf{b}_j$'s, $\mathbf{s}^\top \bar{\mathbf{u}}$, $\mathbf{s}^\top \mathbf{B}_i$'s with random.
- This means that we can simulate $\mathbf{b}_j$'s and the Ψ_i's with uniformly random vectors/matrices, and simulate β with random if $C(x) = 1$, and with $(\widetilde{\mathbf{b}} + \underline{\Psi}_C) \cdot \mathbf{t}$ if $C(x) = 0$.

Extension to multi-bit output. We may proceed as in Section 4.2 to obtain an extension to ℓ-bit output. The modifications are as follows:

- $\mathsf{Compress}$ will compute $\mathbf{A}_{\hat{C}_1}, \ldots, \mathbf{A}_{\hat{C}_\ell}$;
- Enc will compute $\widetilde{\mathbf{b}}$ as before, and sample $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$ to compute $\beta_1, \ldots, \beta_\ell$.