Post-Quantum Secure LFE for L/poly with Smaller Parameters

David Naccache[1], Răzvan Roşie[2], and Lorenzo Spignoli[2]

[1] ENS, Paris, France
david.naccache@ens.fr
[2] University of Luxembourg, Esch-sur-Alzette, Luxembourg
{razvan.rosie,lorenzo.spignoli}@uni.lu

Abstract. Laconic Function Evaluation (LFE) is a recently proposed primitive (FOCS'18), that allows two parties to perform function evaluation in the following way: a "digest" of a circuit representing a function f, is sent by Alice to Bob; next, Bob obtains a ciphertext corresponding to his plaintext m using the digest; finally, Alice, after receiving Bob's ciphertext and having knowledge of her circuit, can decrypt $f(\mathsf{m})$. The protocol is dubbed *laconic* if the sizes of the common reference string, digest and ciphertext are "tiny", and certainly much smaller than the circuit size $|f|$. The inceptive work provides a laconic function evaluation protocol for general circuits under the learning with errors ($\mathsf{LWE}_{n,q,\chi}$) assumption, such that ciphertext and digest grow polynomially with the depth d, but not the size of the circuit. However, the LWE modulus grows as $O(2^{\mathsf{poly}(n,d)})$.

This work puts forward a post-quantum secure and asymptotically *more efficient* laconic function evaluation protocol, but for a *restricted class* of circuits. In particular, we obtain LFE for $\mathsf{L/poly}$ (the class of circuits admitting branching programs of polynomial length) from lattice assumptions. Our LWE modulus belongs to $\widetilde{O}\big(\mathsf{poly}\big(n^7 \cdot 2^{4d}\big)\big)$, such that its size is growing additively with the depth, as opposed to multiplicatively. Our protocol is inspired by an attribute-based encryption scheme proposed by Gorbunov and Vinayagamurthy (AC'15).

Keywords: branching programs, laconic function evaluation, ABE.

1 Introduction

Laconic Function Evaluation (LFE) has been introduced as a novel and forceful primitive in a recent work by Quach *et al.* [QWW18]. Roughly speaking, imagine a scenario where two parties want to evaluate a function f in the following manner: (1) on the one hand – Alice – computes a "digest" of the circuit representation of some function f and sends this digest to the other party involved – Bob; (2) Bob computes a ciphertext CT given two components: the digest of the circuit and his input message m. The ciphertext CT is sent back to Alice; (3) finally, Alice is able to learn the value of $f(\mathsf{m})$ in plain. All these computations are done in a common reference string model, which is proven to be necessary.

The main catalyst behind introducing LFE consists in having protocols that are "Bob-optimized". The challenges posed by the field of "Big Data", where it is often the case that data has to be processed by third parties, draw to attention reliable and practical cryptographic solutions such as secure multi-party computation [GMW87, Yao82a]. Multiple motivating examples are suggested in the introductory article [QWW18], but the key difference that is *emphasized* is the need for a "laconic" protocol: it *must* be the case the digest that Alice sends to Bob is less than the size of the circuit $\mathscr{C}$ representing the function f to be computed.

To give a flavour of such a setting, assume a researcher in neuroscience (Alice) devised a massive program $\mathscr{C}$ that processes images of human brains' magnetic resonance. Such automated investigations, can be used, for instance to detect brain diseases in early stages, using modern machine learning techniques. Through LFE, the researcher will simply publish the digest of $\mathscr{C}$ (which is *short*), and a hospital (managed by Bob), interested in Alice's research, may use the digest to encrypt his magnetic resonance (MR) image repository. Then Alice can decrypt the ciphertext and obtain the results of her program on the dataset that Bob sent. Thus, the patients' personal data would remain private, while Alice would obtain the output of her program.

A second example may look into program patching. If Alice has acquired an operating system (OS) while Bob is the OS developer, it may be the case that from time to time, Bob releases updates for his product. If this is the case, Alice may simply provide a digest for her program that includes the OS source, takes the patch and outputs an updated version of OS. In such a scenario, Bob encrypts the patch through LFE and Alice runs the heavy operation on patching the OS on her side. Clearly, this is advantageous as the size of the program Alice would have sent to the Bob would be large.

Existing work. Quach *et al.* [QWW18] put forth an LFE construction supporting general circuits. Their protocol can be thought as a composition of several steps: first, they achieve an attribute-based laconic function evaluation scheme. Second, they show how to reduce the size of the digest Alice sends to Bob, by employing *laconic oblivious transfer*. Finally, they obtain a scheme for general circuits, by applying the techniques developed in [GKP+13]: Bob encrypts his message through a fully homomorphic encryption (FHE) scheme [Gen09, BGV12, GSW13]; then Bob encrypts the FHE ciphertext via the attribute based laconic function evaluation (step 1), together with the labels of a garbled circuit performing FHE decryption. Essentially, this is similar to [GKP+13], up to the noticeable change of using attribute-based LFE, rather then a plain ABE.

Although QWW18's techniques are clearly interesting, they have to rely on existing works, in order to obtain their main tool – the attribute-based laconic function evaluation for *general circuits*. Namely, they make use of the *public* and *ciphertext* evaluation algorithms introduced by Boneh *et al.* in [BGG+14]. The predicament induced by these powerful evaluation algorithms is the representation *size* of the LWE modulus q, which will be $\mathsf{poly}(\lambda, d)$.

Of independent interest are the relations between LFE and other primitives: the authors of [QWW18] remark that LFE is sufficiently powerful to imply succinct functional encryption[3] (FE). However, the authors leave *open* the question of the converse being true. A second *open* question would consider obtaining adaptive-secure schemes via tight(er) reductions.

Our Contribution. In this work, we provide a lattice-based, post-quantum secure construction of LFE for L/poly circuits – the class of circuits representable through branching programs of polynomial length – that achieves selective security under LWE with polynomial approximation factors[4]. Our construction relies on LWE with asymptotically smaller parameters, when compared to [QWW18]. Our main result may be summarized as follows.

Theorem 1 (LFE for L/poly – Informal). *Let $f : \{0,1\}^k \to \{0,1\}^\ell$ denote a function having its circuit belonging to* L/poly*. Assuming the hardness of* $\mathsf{LWE}_{n,q,\chi}$ *with polynomial approximation factors, there exists a selectively-secure* LFE *protocol for f, having laconic digests, common reference strings and ciphertexts. Moreover, the* LWE *modulus q belongs to* $\widetilde{O}\big(\mathsf{poly}\big(n^7 \cdot 2^{4d}\big)\big)$.

L/poly LFEs from ABEs for Branching Programs. From a technical perspective, we consider different paths for obtaining laconic function evaluation schemes for circuits in L/poly. The approach we propose is somewhat different from existing schemes, as we would like to exploit a different representation.

Consider the branching program representation of some function in L/poly. Assuming the minimal circuit representation of the function has depth d, the branching program will have $L \leq 2^{2d}$ states. At first sight, such a representation seems discouraging. However, some existing attribute-based encryption schemes [GV15, Yam16] that are designed to evaluate branching programs outperform the classical gate-evaluation techniques, ending up with better parameters of ciphertexts.

To give a flavour, consider the gate evaluation ABE algorithms presented in [BGG+14], which form the backbone of [QWW18]. The LWE modulus in [BGG+14] has the size growing as:

$$|q| \approx \log(n) \cdot d \ + \ \text{smaller factors} \ .$$

This is in stark contrast with [GV15], where the LWE modulus grows as:

$$|q| \approx \log(n) + 4 \cdot d \ + \ \text{smaller factors}$$

Therefore, it seems that considering such primitive for branching programs may offer an alternative to classical gate evaluation algorithms.

Paper Organization. In Section 2, we introduce the standard notations to be adopted throughout the paper, followed by the definitions of the primitives

[3] Succinctness means that the size of the functional ciphertext depends on the depth of the supported circuit rather than on its size [GKP+13].

[4] It can also be turned into an adaptively-secure scheme under LWE with sub-exponential approximation factors.

that we use as building blocks. Appendix B reviews the original LFE protocol proposed by Quach *et al.* in [QWW18]. In Sections 3 and 4, we introduce a new LFE scheme for L/poly circuits, and show how to combine it with laconic oblivious transfer in order to achieve a scheme with tiny digests.

2 Preliminaries

Notations. We denote the security parameter by $\lambda \in \mathbb{N}^*$ and we assume it is implicitly given to all algorithms in the unary representation 1^λ. An algorithm is considered equivalent to a Turing machines and is supposed to be randomized; "Probabilistic polynomial-time" in the security parameter is denoted by PPT. Given some (randomized) algorithm $\mathcal{A}$, the action of running $\mathcal{A}$ on input(s) $(1^\lambda, x_1, \dots)$ with uniform random coins r and assigning the output(s) to $(y_1, \dots)$, is denoted by $(y_1, \dots) \leftarrow_\$ \mathcal{A}(1^\lambda, x_1, \dots ; r)$. When $\mathcal{A}$ is given oracle access to some procedure $\mathcal{O}$, we write $\mathcal{A}^{\mathcal{O}}$. For a finite set S, we denote its cardinality by $|S|$ and the action of sampling an element x uniformly at random from X by $x \leftarrow_\$ X$. Bold variables such as $\mathbf{w}$ represent column vectors, while, bold capitals stand for matrices (e.g. $\mathbf{A}$). A subscript $\mathbf{A}_{i,j}$ refers to entry (i, j) in the matrix. For any variable $k \in \mathbb{N}^*$, we define $[k] \coloneqq \{1, \dots, k\}$. A real-valued function $\text{NEGL}(\lambda)$ is negligible if $\text{NEGL}(\lambda) \in \mathcal{O}(\lambda^{-\omega(1)})$. We denote the set of all negligible functions by NEGL. Throughout the paper $\perp$ stands for a special error symbol. We use $||$ to denote concatenation. For completeness, we recall standard algorithmic and cryptographic primitives to be used. We consider circuits as the prime model of computation for representing (abstract) functions. Unless stated otherwise, we use k to denote the input length of the circuit and d for its depth.

2.1 Cryptographic Assumptions

The Learning With Error problem, proposed by Regev in [Reg05], requires one to find the secret vector $\mathbf{s}$ over $\mathbb{F}_q^\ell$, given a polynomially many relations of the form $(\mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{e})$; here $\mathbf{A}$ denotes a randomly sampled matrix over $\mathbb{F}_q^{k\times\ell}$, while $\mathbf{e} \in \mathbb{F}^k$ is a small error term sampled from an appropriate distribution χ. Roughly, the decision version of the problem, asks to distinguish between the aforementioned distribution as opposed to the uniform one.

Definition 1 (Decisional LWE). *The advantage of any* PPT *adversary* Adv *in distinguishing between the following two distributions is negligible:*

$$\mathsf{Adv}^{\mathsf{LWE}}_{\mathcal{A}}(\lambda) \coloneqq \Big| \Pr\left[1 \leftarrow \mathsf{Adv}\big(1^\lambda, \mathbf{A}, \mathbf{u}\big)\right] - \Pr\left[1 \leftarrow \mathsf{Adv}\big(1^\lambda, \mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{e}\big)\right] \Big| \in \text{NEGL}(\lambda)$$

where $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{k\times\ell}$, $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^\ell$, $\mathbf{e} \leftarrow_\chi \mathbb{Z}_q^k$ *while* $\mathbf{u} \leftarrow_\$ \mathbb{Z}_q^k$ *is a randomly sampled vector.*

In our work, we will make use of the following standard result:

Lemma 1 (Smudging Lemma). *Let B, B' denote two integer bounds, modelled as polynomials in some parameter λ. Let $\delta \in [-B, B]$ and $\gamma \in [-B', B']$ be sampled uniformly at random. Then the distribution of γ is statistically indistinguishable from that of $\delta + \gamma$ as long as B/B' is negligble in λ.*

2.2 Standard Primitives

Branching Programs. A branching program corresponds to a sequential representation of a circuit. One should imagine it as a layered graph, the arcs between two layers representing the possible transitions between consecutive states (or levels). A branching program is evaluated from a starting node which has two emerging paths. Depending on the value of some input bit, one of the two paths is chosen. The process is repeated at any given step, until a terminal state is reached. In a celebrated result, [Bar89] has shown how to convert any circuit in NC^1 into a branching program having a polynomial number of states – herein denoted by L – and a width fixed to 5. We defined them below.

Definition 2 (Branching Programs). *Let $f : \{0,1\}^k \to \{0,1\}^\ell$ be a function whose circuit representation belongs to NC^1. A branching program BP for f consists of L tuples of the form:*

$$\left(\{\mathsf{var(i)}, \pi_{i,0}, \pi_{i,1}\}_{i\in[L]}\right) \ ,$$

where $\mathsf{var} : [L] \to [k]$ denotes a function that associates a state with some input bit, while $\{\pi_{i,0}, \pi_{i,1}\}_{i\in[L]}$ are two permutations that are used to evaluate the branching program. The evaluation of the branching program is done according to:

$$b \leftarrow \pi_{L,x_{\mathsf{var}(L)}} \circ \pi_{L-1,x_{\mathsf{var}(L-1)}} \circ \ldots \circ \pi_{1,x_{\mathsf{var}(1)}}(\sigma_0)$$

and is equivalent with the output of $f(x)$ for some $x \in \{0,1\}^k$. σ_0 denotes the initial state $(1,0,0,0,0)$.

Fully-homomorphic encryption [GSW13] and garbling schemes [Yao82b], are considered standard primitives and not defined or reviewed herein.

2.3 Attribute-Based Encryption

An attribute-based encryption scheme (ABE) (in the *key-policy* setting) is an encryption scheme where a key is generated for a Boolean predicate P, while a ciphertext is the encryption of a set of attributes α and of some message. Thus, the owner of the key can recover the secret message encrypted together with attributes α if $P(\alpha) = 1$, or nothing otherwise.

Definition 3 (ABE [GPSW06]). *A key-policy attribute-based encryption scheme is a tuple of* PPT *algorithms such that:*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow_\$ \mathsf{Setup}(1^\lambda)$*: takes as input the unary representation of the security parameter λ and outputs the master public key mpk and a master secret key msk.*
- $\mathsf{sk}_P \leftarrow_\$ \mathsf{KeyGen}(\mathsf{msk}, P)$*: given the master secret key and a policy P, the (potentially randomized) key-derivation outputs a corresponding sk_P.*
- $\mathsf{CT} \leftarrow_\$ \mathsf{Enc}(\mathsf{mpk}, \alpha, \mathsf{M})$*: the randomized encryption procedure encrypts the plaintext M with respect to some attribute set α.*

– $\mathsf{Dec}(\mathsf{sk}_P, \mathsf{CT})$*: decrypts the ciphertext* CT *using the key* sk_P *and obtains* M *if* $P(\alpha) = 1$ *or a special symbol* $\perp$*, in case the decryption procedure fails (i.e.* $P(\alpha) = 0$*).*

We say that an ABE *satisfies correctness if for all* $\mathsf{m} \in \mathcal{M}$*, for all predicates* P *and for all attributes* α *we have:*

$$\Pr\left[y = \mathsf{m} \;\middle|\; \begin{array}{l} (\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{ABE.Setup}(1^\lambda) \wedge \\ \mathsf{sk}_f \leftarrow_{\$} \mathsf{ABE.KeyGen}(\mathsf{msk}, P) \wedge \\ \mathsf{CT} \leftarrow_{\$} \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{m}, \alpha) \wedge \\ y \leftarrow \mathsf{ABE.Dec}(\mathsf{CT}, \mathsf{sk}_P) \wedge P(\alpha) = 1 \end{array}\right] = 1 \ .$$

We say an attribute-based encryption is selectively secure if the advantage of any PPT *adversary* $\mathcal{A}$ *in winning the following game is negligible:* $\mathsf{m} \in \mathcal{M}$*, for all predicates* P *and for all attributes* α *we have:*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} \alpha^* \leftarrow_{\$} \mathcal{A}(1^\lambda, 1^k, 1^d) \\ (\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{ABE.Setup}(1^\lambda, 1^k, 1^d) \wedge \\ (\mathsf{m}_0, \mathsf{m}_1) \leftarrow_{\$} \mathcal{A}(\mathsf{mpk}) \wedge \\ b \leftarrow_{\$} \{0,1\} \wedge \\ \mathsf{sk}_f \leftarrow_{\$} \mathcal{A}^{\mathsf{KeyGen}_{\mathsf{msk}}(\cdot)}(\mathsf{mpk}) \wedge \\ \mathsf{CT}^*_\alpha \leftarrow_{\$} \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{m}_b, \alpha^*) \wedge \\ b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}_{\mathsf{msk}}(\cdot)}(\mathsf{mpk}, \mathsf{CT}^*_\alpha) \end{array}\right] = 1 \ .$$

The only restriction that we impose is that $\mathcal{A}$ *does not query for keys corresponding to* $\{P : P(\alpha^*) = 1\}$*.*

$\underline{\mathsf{sFULL\text{-}SIM\text{-}LFE}^{\mathcal{A}}_{\mathsf{LFE}}(\lambda)}$:	$\underline{\mathsf{FULL\text{-}SIM\text{-}LFE}^{\mathcal{A}}_{\mathsf{LFE}}(\lambda)}$:
$b \leftarrow_{\$} \{0,1\}$	$b \leftarrow_{\$} \{0,1\}$
$(\mathsf{m}^*, \mathscr{C}, k, d) \leftarrow_{\$} \mathcal{A}(1^\lambda)$	$(k, d) \leftarrow_{\$} \mathcal{A}(1^\lambda)$
$\mathsf{crs} \leftarrow_{\$} \mathsf{LFE.crsGen}(1^\lambda, 1^k, 1^d)$	$\mathsf{crs} \leftarrow_{\$} \mathsf{LFE.crsGen}(1^\lambda, 1^k, 1^d)$
$\mathsf{digest}_{\mathscr{C}} \leftarrow_{\$} \mathsf{LFE.Compress}(\mathsf{crs}, \mathscr{C})$	$(\mathsf{m}^*, \mathscr{C}) \leftarrow_{\$} \mathcal{A}(\mathsf{crs})$
if $b = 0$:	$\mathsf{digest}_{\mathscr{C}} \leftarrow_{\$} \mathsf{LFE.Compress}(\mathsf{crs}, \mathscr{C})$
$\quad \mathsf{CT}^* \leftarrow_{\$} \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, \mathsf{m}^*)$	if $b = 0$:
else	$\quad \mathsf{CT}^* \leftarrow_{\$} \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, \mathsf{m}^*)$
$\quad \mathsf{CT}^* \leftarrow_{\$} \mathcal{S}(\mathsf{crs}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathscr{C}(\mathsf{m}^*))$	else
$b' \leftarrow_{\$} \mathcal{A}(\mathsf{crs}, \mathsf{CT}^*)$	$\quad \mathsf{CT}^* \leftarrow_{\$} \mathcal{S}(\mathsf{crs}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathscr{C}(\mathsf{m}^*))$
return $b = b'$	$b' \leftarrow_{\$} \mathcal{A}(\mathsf{CT}^*)$
	return $b = b'$

Fig. 1. Left: the selective simulation security experiment $\mathsf{sFULL\text{-}SIM\text{-}LFE}$ defined for a laconic function evaluation scheme LFE. Right: its adaptive version.

2.4 Laconic Function Evaluation

Definition 4 (Laconic Function Evaluation [QWW18]). *Let* LFE *denote a laconic function evaluation scheme for a class of circuits* $\mathfrak{C}_\lambda$*. It consists of four algorithms* (crsGen, Compress, Enc, Dec)*:*

- $\mathsf{crs} \leftarrow_\$ \mathsf{LFE.crsGen}(1^k, 1^d, 1^\lambda)$*: assuming the input size and the depth of the circuit in the given class are* k *and* d*, a common reference string* crs *of appropriate length is generated. We assume that* crs *is implicitly given to all algorithms.*
- $\mathsf{digest}_{\mathscr{C}} \leftarrow_\$ \mathsf{LFE.Compress}(\mathsf{crs}, \mathscr{C})$*: the compression algorithm takes a description of the circuit* $\mathscr{C}$ *and produces a digest* $\mathsf{digest}_{\mathscr{C}}$*.*
- $\mathsf{CT} \leftarrow_\$ \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, \mathsf{m})$*: takes as input the message* m *as well as the digest of* $\mathscr{C}$ *and produces a ciphertext* CT*.*
- $\mathsf{LFE.Dec}(\mathsf{crs}, \mathsf{CT}, \mathscr{C})$*: if the parameters are correctly generated, the decryption procedure recovers* $\mathscr{C}(\mathsf{m})$*, given the ciphertext encrypting* m *and circuit* $\mathscr{C}$ *or a special symbol* $\perp$*, in case the decryption procedure fails.*

We require the LFE *scheme to achieve the following properties:*

- ***Correctness*** *- for all* $\mathscr{C} : \{0,1\}^k \to \{0,1\}^\ell$ *of depth* d *and for all* $\mathsf{m} \in \{0,1\}^k$ *we have:*

$$\Pr\left[y = \mathscr{C}(\mathsf{m}) \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow_\$ \mathsf{LFE.crsGen}(1^\lambda, 1^k, 1^d) \wedge \\ \mathsf{digest}_{\mathscr{C}} \leftarrow_\$ \mathsf{LFE.Compress}(\mathsf{crs}, \mathscr{C}) \wedge \\ \mathsf{CT} \leftarrow_\$ \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, \mathsf{m}) \wedge \\ y \leftarrow \mathsf{LFE.Dec}(\mathsf{crs}, \mathscr{C}, \mathsf{CT}) \end{array} \right] = 1 \ .$$

- ***Security****: there exists a* PPT *simulator* $\mathcal{S}$ *such that for any stateful* PPT *adversary* $\mathcal{A}$ *we have:*

$$\mathsf{Adv}^{\mathsf{FULL\text{-}SIM\text{-}LFE}}_{\mathcal{A},\mathsf{LFE}}(\lambda) := \left| \Pr[\mathsf{FULL\text{-}SIM\text{-}LFE}^{\mathcal{A}}_{\mathsf{LFE}}(\lambda) = 1] - \frac{1}{2} \right|$$

  is negligible, where FULL-SIM-LFE *is defined in Figure 1 (right side).*

  A relaxed version of the security experiment, denoted sFULL-SIM-LFE *asks the adversary to provide the challenge message and function in the beginning of the security experiment.*

- ***Laconic outputs****: As per [QWW18, p13-14], we require the size of digest to be laconic* $|\mathsf{digest}_{\mathscr{C}}| \in O(poly(\lambda))$ *and we impose succinctness constraints for the sizes of the ciphertext and public parameters.*

Remark 1 (Comparison between the security definitions of FE *and* LFE*.).* The security definition depicted in Figure 1 seem very close to the one of a functional encryption scheme (see for example [GKP+13]), but one has to notice a conceptual difference. A functional encryption scheme should be imagined as involving three distinct entities: a trusted third party that generates parameters

and issues functional keys, the encrytor, and the decryptor that has to operate on correctly generated ciphertexts and functional keys. On the other hand, a laconic function evaluation protocol has only two parties[5], which imposes some restrictions: Alice has to generate the digest corresponding to some f correctly[6], and the ciphertext Bob generates is expected to be correct.

3 LFEs for L/poly with Smaller Parameters

Motivation - Reducing Ciphertext's Size. The existing laconic function evaluation constructions that target general circuits have the size of the resulting parameters dependent by the depth d of the circuits to be evaluated. A natural question that remains is to optimize the prior result, by further reducing the dimensions of digest and/or ciphertext. Considering circuits in $\mathsf{L/poly}$, the answer is positive, although such a transform is not straightforward.

Gorbunov and Vinayagamurthy observe in [GV15] that evaluating branching programs (Definition 2) of length L, and using custom-made evaluation $\mathsf{Eval}_{\mathsf{PK}}$ and $\mathsf{Eval}_{\mathsf{CT}}$ algorithms induces a reduced noise level, up to the point that $q = \widetilde{O}\big(\mathsf{poly}\big(n^7 \cdot L^2\big)\big)$, where q, n, χ are parameters of the Learning with Errors problem. Put differently, the *size* of the modulus q has the following form:

$$7 \cdot \log(n) + 2 \cdot \log(L) + \underbrace{\log\left(7 \cdot \log(n) + 2 \cdot \log(L)\right) + \log(\text{const})}_{\text{smaller factor}}$$

Assuming that $L \leq 4^d$, where d is the depth of the circuit, and ignoring the double-log part, we can approximate the size of q by:

$$\begin{aligned} |q| \approx & 7 \cdot \log(n) + 2 \cdot \log(L) \\ |q| \approx & 7 \cdot \log(n) + 4 \cdot d \end{aligned} \tag{1}$$

Clearly, when comparing the size of the LWE modulus in Equation (1) to the one [QWW18] we observe its magnitude grows additively with the depth, as opposed to multiplicatively. A second notable thing is the fact that evaluating branching programs requires encrypting a small *state*. That is beneficial, as the overhead an attribute-based laconic function evaluation scheme will add to the FHE ciphertext[7] is slightly small.

Remark 2. Even more impressive is the result of Yamada [Yam16], that shows the construction of Gorbunov can evaluate unbounded length branching programs while maintaining a *compact* ciphertext under standard LWE (with polynomial approximation factors).

Appendix B reviews the main scheme in [QWW18]. In what follows, we describe an attribute-based laconic function evaluation ($\mathsf{AB\text{-}LFE}$) scheme, whose *definition* and *security experiments* are in Appendix B.1.

[5] We ignore the parameter generation step.
[6] This is equivalent with an honest key generation process in FE.
[7] FHE is needed to support full circuits

3.1 Attribute-Based LFE for L/poly

We start with a simple construction of an attribute-based laconic function evaluation, which follows immediately from the scheme in [GV15]. The main idea is to release LWE encodings of attributes, as well as 5 matrices that are used in the evaluation of the branching program[8]. The message is going to be blinded separately. The scheme follows the intuition described in Section 1.

Definition 5 (Attribute-Based LFE for $\mathsf{L/poly}$). *Let $\mathcal{C}_{d,k,\ell}$ denote a class of boolean circuits in $\mathsf{L/poly}$, taking as input k bits, ℓ output bits and having depth d, and let $\mathcal{BP}_L$ denote the corresponding class of branching programs parametrized by length at most L and width at most 5. Let LWE be the learning with errors problem parametrized by dimension $n = \mathsf{poly}(\lambda)$, a modulus $q = \widetilde{O}(n^7 \cdot L^2)$, $m = \Omega(n \cdot \log(q))$ and some noise distribution χ. Our attribute-based laconic evaluation function $\mathsf{AB\text{-}LFE}$ consists of the following algorithms:*

$\mathsf{AB\text{-}LFE.crsGen}(1^\lambda, 1^k, 1^d)$**:**

For every input bit $i \in [k]$, sample uniformly at random a matrix $\mathbf{A}_i$ over $\mathbb{Z}_q^{n\times m}$:

$$\mathbf{A}_i \leftarrow_\$ \mathbb{Z}_q^{n\times m} \ .$$

Furthermore, sample $\mathbf{A}^c \leftarrow \mathbb{Z}_q^{n\times m}$, as well as 5 matrices to encode a state of the branching program:

$$\mathbf{V}_{0,i} \leftarrow_\$ \mathbb{Z}_q^{n\times m} \ , \forall i \in [5].$$

Return $\mathsf{crs} \leftarrow \left(\mathbf{A}^c, \{\mathbf{A}_1, \ldots, \mathbf{A}_k\}, \{\mathbf{V}_{0,1}, \mathbf{V}_{0,2}, \mathbf{V}_{0,3}, \mathbf{V}_{0,4}, \mathbf{V}_{0,5}\}\right)$.

$\mathsf{AB\text{-}LFE.Compress}(\mathsf{crs}, \mathscr{C})$**:**

Let $\{\mathsf{BP}_j\}_{j\in[\ell]}$ denote the branching programs corresponding to the circuit representation $\mathscr{C}$ of some $f : \{0,1\}^k \to \{0,1\}^\ell$. Compute:

$$\mathbf{V}_{\mathsf{BP}_j} \leftarrow \mathsf{Eval}_{\mathsf{PK}}\left(\mathsf{BP}_j, \mathbf{A}^c, \{\boldsymbol{A}_i\}_{i\in[k]}, \{\boldsymbol{V}_{0,i}\}_{i\in[5]}\right) \ , \forall\ j \in [\ell] \ ,$$

using $\mathsf{Eval}_{\mathsf{PK}}$ as defined in [GV15]. Set $\mathsf{digest}_{\mathscr{C}} \leftarrow \{\mathbf{V}_{\mathsf{BP}_j}\}_{j\in[\ell]}$ and return it.

$\mathsf{AB\text{-}LFE.Enc}\left(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\alpha, \mathsf{m})\right)$**:**

Sample $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^m, \mathbf{e}_i \leftarrow_\$ \chi^m$ for each $i \in [k]$ and compute:

$$\vec{b}_i \leftarrow \mathbf{s}^\top \cdot (\boldsymbol{A}_i - \alpha_i \cdot \boldsymbol{G}) + \mathbf{e}_i^\top \ ,$$

where $\boldsymbol{G}$ is the Matrix Gadget [GMP19, GSW13].

Let the initial BP state vector be $\mathbf{v}_0 \leftarrow (1,0,0,0,0)$. Let the noise terms $\mathbf{e}_{0,i} \leftarrow_\chi \chi$. For each $i \in [5]$ sample the ciphertexts corresponding to the first level of the branching program:

$$\mathbf{v}_{0,i} \leftarrow \mathbf{s}^\top \cdot (\mathbf{V}_{0,i} + v_i \cdot \mathbf{G}) + \mathbf{e}_{0,i} \ , \ \textit{where } v_i \textit{ is the } i^{th} \textit{ bit of } \mathbf{v}.$$

[8] The factor 5 is due to the evaluation of the branching program.

Compute the auxiliary ciphertext:

$$\vec{b}^c \leftarrow \mathbf{s}^\top \cdot (\mathbf{A}^c + \mathbf{G}) + \mathbf{e}^c$$

Parse the digest as $\{\mathbf{V}_{\mathsf{BP}_j}\}_{j\in[\ell]}$ *and encode* $\mathsf{m} \coloneqq (\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$ *as:*

$$\vec{b}_{\mathsf{m},j} \leftarrow \mathbf{s}^\top \cdot \left(\mathbf{V}_{\mathsf{BP}_j} + \mathbf{G}\right) + \mathbf{e}_{\mathsf{m},j} + \lfloor q/2 \rceil \cdot \mathsf{m}_j$$

$\underline{\mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, \mathsf{BP}, \mathsf{digest}_{\mathscr{C}}, \mathsf{CT}) :}$

Compute

$$\vec{r}_j \leftarrow \mathsf{Eval}_{\mathsf{CT}}\left(\mathsf{BP}_j, \{\mathbf{A}_i\}_{i\in[k]}, \mathbf{A}^c, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \{\vec{b}_i\}_{i\in[k]}, \vec{b}^c, \{\vec{v}_{0,i}\}_{i\in[5]}\right) \ . \quad (2)$$

By the correctness of the underlying ABE *scheme, Equation* (2) *is equivalent with (if* $\mathsf{BP}_j(\alpha) = 1$*)*

$$\mathbf{s}^\top \cdot \left(\mathbf{V}_{\mathsf{BP}_j} + \mathbf{G}\right) + \mathbf{e}_{\mathsf{BP}_j} \ . \quad (3)$$

Subtract from $\vec{b}_{\mathsf{m},j}$ *the vector* $\vec{r}_j$ *and obtain:*

$$\mathbf{e}_{\mathsf{m},j} - \mathbf{e}_{\mathsf{BP}_j} + \lfloor q/2 \rceil \cdot \mathsf{m}_j \ .$$

Remove the noise and recover m_j*.*

Proposition 1 (Correctness). *The construction in Definition 5 is correct.*

Proof (Proposition 1). By the correctness of the underlying ABE scheme, the correctness of our scheme follows trivially. We iterate over al $j \in [\ell]$ the following steps. First, we invoke the correctness of the $\mathsf{Eval}_{\mathsf{CT}}$ algorithm in [GV15], in order to end up with $\vec{r}_j$ in Equation (2):

Then, we are left with $\mathbf{e}_{\mathsf{m},j} - \mathbf{e}_{\mathsf{BP}_j} + \lfloor q/2 \rceil \cdot \mathsf{m}_j$. The precondition is that the noise generated by $\mathsf{Eval}_{\mathsf{CT}}$ – namely $\mathbf{e}_{\mathsf{BP}_j}$ – is small. Therefore, we can subtract $\mathbf{r}$ from $\mathbf{b}_{\mathsf{m},j}$ and extract the noise. □

Theorem 2 (Security for AB-LFE for Branching Programs). *Under the* $\mathsf{LWE}_{n,q,\chi}$ *assumption with polynomial approximation factors, the scheme in Definition 5 is a selectively-secure attribute-based laconic function evaluation scheme (Definition 8).*

Proof. First, we describe the internal working of our simulator. Then we show how the ciphertext can be simulated via a hybrid argument, by describing the hybrid games and their code. Third, we prove the transition between each consecutive pair of hybrids.

Simulator. Our simulator $\mathcal{S}_{\mathsf{AB\text{-}LFE}}$ is given the digest $\mathsf{digest}_{\mathscr{C}}$, the circuit $\mathscr{C}$, the value of $\mathscr{C}(\alpha^*)$ (i.e. m^*), together with crs. Given that we are in an attribute-based setting, the value $\mathscr{C}(\alpha^*)$ can be either $\perp$ – in which case the simulation is trivial – or an actual value to be simulated (m^*). Henceforth, $\mathcal{S}_{\mathsf{AB\text{-}LFE}}$ proceeds as follows: (1) samples all LWE encodings uniformly at random; (2) replaces the component $\vec{b}_{\mathsf{m}}$ with a surrogate.

We use a hybrid argument, and show the transitions between hybrids below.

Game_0**:** this is the real LFE experiment adapted to the attribute-based setting. $\mathsf{Game}_{1.0}$ is identical to Game_0.

$\mathsf{Game}_{1.j}$**:** in this game, we change the distribution of each $\vec{b}_{\mathsf{m},j}$. To this end, we rely on a smudging lemma introduced in [AJL+12], in order to add to each $\vec{b}_{\mathsf{m},j}$ encrypting challenge m^*_j, an extra noise component $\mathbf{e}_{\mathsf{BP}_j}$.

Game_2**:** finally, we change all components that are used to encrypt the state through a reduction to LWE.

Claim ($\mathsf{Game}_{1.0} \to \mathsf{Game}_{1.1}$*).* The distance between $\mathsf{Game}_{1.0}$ and $\mathsf{Game}_{1.1}$ is statistically close to 0.

Proof. We change the distribution of $\vec{b}_{\mathsf{m},j}$ from:

$$\vec{b}_{\mathsf{m},j} \leftarrow \mathbf{s}^\top \cdot \left(\mathbf{V}_{\mathsf{BP}_j} + \mathbf{G}\right) + \mathbf{e}_{\mathsf{m},j} + \lfloor q/2 \rceil \cdot \mathsf{m}^*_j$$

to

$$\vec{b}_{\mathsf{m},j} \leftarrow \mathbf{s}^\top \cdot \left(\mathbf{V}_{\mathsf{BP}_j} + \mathbf{G}\right) + \mathbf{e}_{\mathsf{m},j} - \mathbf{e}_{\mathsf{BP}_j} + \lfloor q/2 \rceil \cdot \mathsf{m}^*_j$$

This change is possible thanks to the smudging lemma (Lemma 1). □

The previous argument is used to justify every transition, for $j \in [\ell]$.

Claim ($\mathsf{Game}_{1.\ell} \to \mathsf{Game}_2$*).* The distance between $\mathsf{Game}_{1.\ell}$ and Game_2 is bounded by the advantage against $\mathsf{LWE}_{n,q,\chi}$ with polynomial approximation factors.

Proof. The LWE game provides us with either correctly generated LWE tuples (as per $\mathsf{Game}_{1.\ell}$) or with uniformly sampled elements (Game_2). The simulator uses the given elements in order to simulate the following ciphertext components:

$$\left(\{\vec{b}_i\}_{i\in[k]}, \{\vec{v}_{0,i}\}_{i\in[5]}, \vec{b}^c\right)$$

Note that the simulation is always possible given some vector $\vec{u}$ (which is either a proper LWE sample or random): the simulator can interpret the public matrices constituting the crs, being of form $\mathbf{A}_i - \alpha_i \cdot \mathbf{G}$ or $\mathbf{V}_i + v_i \cdot \mathbf{G}$. Note that decryption is possible due to the previous game hop and to the additional noise component. This setting corresponds to a fully simulated ciphertext. □

Finally, by applying the union bound, we can conclude that the AB-LFE is selectively secure under LWE with polynomial approximation factors. □

4 LFE for Branching Programs in L/poly

We introduce our LFE construction for L/poly built on top of the multi-bit *attribute-based* LFE introduced previously. As per [QWW18, GKP+13], for simplicity, we use of a two-outcome attribute based laconic function evaluation scheme, which can be obtained generically from an AB-LFE. Its functionality can be summarized as follows:

$$\mathsf{AB\text{-}LFE}_2.\mathsf{Dec}(\mathsf{crs}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathsf{AB\text{-}LFE}_2.\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\alpha, \mathsf{m}_0, \mathsf{m}_1)) = \mathsf{m}_{\mathscr{C}(\alpha)} \ .$$

That is, the decryption returns one out of two messages, as pointed out by the output bit of $\mathscr{C}(\mathsf{m})$. We define the scheme below, and postpone its *correctness* and *security* analysis to Appendix A.

Definition 6 (LFE for L/poly with Succinct Parameters). *Let* AB-LFE *denote the attribute-based laconic function evaluation scheme for* L/poly *circuits of depth d, presented in Section 3.1. Let* FHE *stand for a semantic secure fully homomorphic encryption scheme and let* GS *denote a semantic secure garbling scheme. Let the length of a message be denoted by k and the length of an* FHE *ciphertext encrypting k bits be t. Then* $\overline{\mathsf{LFE}}$ *stands for a laconic function evaluation scheme for* L/poly *having succinct ciphertexts.*

- $\overline{\mathsf{crs}} \leftarrow_{\$} \overline{\mathsf{LFE}}.\mathsf{crsGen}(1^\lambda, 1^k, 1^d)$*: the* $\overline{\mathsf{crs}}$ *is instantiated via* $\mathsf{AB\text{-}LFE}_2.\mathsf{Setup}$

$$\mathsf{crs}_{\mathsf{AB\text{-}LFE}} \leftarrow_{\$} \mathsf{AB\text{-}LFE}_2.\mathsf{Setup}(1^\lambda, 1^k, 1^d) \ .$$

  Set $\overline{\mathsf{crs}} \leftarrow \mathsf{crs}_{\mathsf{AB\text{-}LFE}}$ *and return it.*

- $\mathsf{digest}_{\mathscr{C}} \leftarrow \overline{\mathsf{LFE}}.\mathsf{Compress}(\overline{\mathsf{crs}}, \mathscr{C})$*: run* $\mathsf{AB\text{-}LFE}_2.\mathsf{Compress}$ *and return its output*[9] *as* $\mathsf{digest}_{\mathscr{C}}$*:*

$$\mathsf{digest}_{\mathscr{C}} \leftarrow \mathsf{AB\text{-}LFE}_2.\mathsf{Compress}(\mathsf{crs}_{\mathsf{AB\text{-}LFE}}, \mathsf{FHE.Eval}(\mathscr{C}, \cdot)) \ .$$

- $\mathsf{CT} \leftarrow_{\$} \overline{\mathsf{LFE}}.\mathsf{Enc}(\overline{\mathsf{crs}}, \overline{\mathsf{digest}_{\mathscr{C}}}, \mathsf{m})$*: the encryption algorithm first samples* FHE *keys* $(\mathsf{hpk}, \mathsf{hsk})$ *and encrypts* m*:*

$$\mathsf{CT}_{\mathsf{FHE}} \leftarrow_{\$} \mathsf{FHE.Enc}(\mathsf{hpk}, \mathsf{m}) \ ,$$

  such that $|\mathsf{CT}_{\mathsf{FHE}}| = t$.

  Consider $\mathscr{C}_{aux}(\cdot)$ *that hardcodes* hsk*, takes as input an* FHE *ciphertext and returns* $\mathsf{FHE.Dec}_{\mathsf{hsk}}(\cdot)$*. Garble* $\mathscr{C}_{aux}$ *and obtain:*

$$(\Gamma, \{L_i^0, L_i^1\}_{i=1}^t) \leftarrow_{\$} \mathsf{GS.Garble}(\mathscr{C}_{aux}) \ .$$

  Use $\mathsf{AB\text{-}LFE}_2$ *to encrypt the ciphertext and each label* L_i^b*, for all* $i \in [t]$ *and* $b \in \{0,1\}$*:*

$$\overline{\mathsf{CT}} \leftarrow \mathsf{AB\text{-}LFE}_2.\mathsf{Enc}(\mathsf{crs}_{\mathsf{AB\text{-}LFE}}, \mathsf{digest}_{\mathscr{C}}, (\mathsf{CT}_{\mathsf{FHE}}, \{L_{i,0}, L_{i,1}\}_{i\in[t]}))$$

  The ciphertext CT *is set to be the tuple* $(\Gamma, \mathsf{CT}_{\mathsf{FHE}}, \overline{\mathsf{CT}})$.

- $\overline{\mathsf{LFE}}.\mathsf{Dec}(\overline{\mathsf{crs}}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathsf{CT})$*: Recover the labels corresponding to* $\mathsf{FHE.Enc}(\mathsf{hpk}, \mathsf{BP}(\mathsf{m}))$*:*

$$\{L_i\}_{i\in[t]} \leftarrow \mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}_{\mathsf{AB\text{-}LFE}}, \mathsf{digest}_{\mathscr{C}}, \mathsf{BP}, \mathsf{CT})$$

  Return $\mathsf{GS.Dec}\,(\Gamma, \{L_1, \ldots, L_t\})$.

[9] Internally, $\mathscr{C}$ is represented as a sequence of branching programs.

References

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, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

Bar89. David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

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, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

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, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Heidelberg, August 2017.

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

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 ACM STOC*, pages 555–564. ACM Press, June 2013.

GMP19. Nicholas Genise, Daniele Micciancio, and Yuriy Polyakov. Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 655–684. Springer, Heidelberg, May 2019.

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

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

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, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

GV15. Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 550–574. Springer, Heidelberg, November / December 2015.

QWW18. Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Yam16. Shota Yamada. Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 32–62. Springer, Heidelberg, May 2016.

Yao82a. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

Yao82b. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.

A Analysis of the Scheme in Section 4

Proposition 2 (Correctness). *The laconic function evaluation scheme in Definition 6 is correct.*

Proof. The correctness follows immediately from the correctness of $\mathsf{AB\text{-}LFE}_2$ and GS. First, we get the labels $L_i^{x_i}$, where $x_i \leftarrow \mathsf{FHE.Eval}(\mathscr{C}, \mathsf{FHE.Enc}(\mathsf{hpk}, \mathsf{m}))$. Then, via Yao's garbling scheme we get $\mathscr{C}(\mathsf{m})$. □

Simulation security follows through a hybrid argument.

Theorem 3 (Security). *Let $\mathfrak{C}_{k,d}$ denote a class of circuits and let $\overline{\mathsf{LFE}}$ be the laconic function evaluation scheme from Definition 6. Let GS and FHE denote a garbling, respectively fully homomorphic encryption schemes. Then $\overline{\mathsf{LFE}}$ is an adaptively secure laconic function evaluation scheme for $\mathsf{L/poly}$ under LWE with polynomial approximation factors. Moreover, the advantage of any* PPT *bounded adversary $\mathcal{A}$ against the adaptive simulation security of the $\overline{\mathsf{LFE}}$ scheme is bounded as follows:*

$$\mathsf{Adv}^{\mathsf{sFULL\text{-}SIM\text{-}LFE}}_{\overline{\mathsf{LFE}},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{FULL\text{-}SIM\text{-}GS}}_{\mathsf{GS},\mathcal{R}_1}(\lambda) + \mathsf{Adv}^{\mathsf{FULL\text{-}SIM\text{-}FHE}}_{\mathsf{FHE},\mathcal{R}_2}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{sFULL\text{-}SIM\text{-}ABLFE}}_{\mathsf{AB\text{-}LFE},\mathcal{R}_3}(\lambda) \ .$$

Proof. **Simulator.** We describe the execution of $\mathcal{S}_{\overline{\mathsf{LFE}}}$: first, it runs the simulator corresponding to FHE ciphertext; second, it executes the simulator of the garbled circuit $\mathcal{S}_{\mathsf{GS}}$; finally, it runs the simulator of the underlying attribute-based LFE scheme in order to obtain the bulk of the ciphertext.

The proof requires a hybrid argument. We present below the games and the transitions between them:

Game_0**:** corresponds to the FULL-SIM-LFE game where b is set to 0.

Game_1**:** we change the bulk ciphertext to the one obtained by the $\mathcal{S}_{\mathsf{AB\text{-}LFE}}$ simulator, the distance to the previous game being bounded by the advantage in Theorem 2.

Game_2**:** in this game, we switch to the usage of the garbled scheme simulator GS in order to compute the labels corresponding to the second part of the ciphertext. The game hop is bounded by the simulation security of the garbling scheme.

Game_3**:** we change the underlying FHE ciphertext, relying on its semantic security.

Claim (Transition between Game_0 *and* Game_1*).* The advantage of any PPT adversary in distinguishing between Game_0 and Game_1 is bounded as follows:

$$\mathsf{Adv}^{\mathsf{Game}_0 \to \mathsf{Game}_1}_{\mathcal{A}_1}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{sFULL\text{-}SIM\text{-}ABLFE}}_{\mathsf{AB\text{-}LFE},\mathcal{B}_1}(\lambda) \ .$$

Proof ($\mathsf{Game}_0 \to \mathsf{Game}_1$*).* Our proof relies on the security of AB-LFE scheme in order to switch elements of the first component of the ciphertext to simulated ones. The reduction $\mathcal{B}_1$ gets some message m, together with a circuit $\mathscr{C}$, from the sFULL-SIM-LFE adversary $\mathcal{A}_1$. $\mathcal{B}_1$ also samples the FHE keys, constructs the FHE ciphertext, and garbles the FHE decryption circuits. These quantities are sent to the sFULL-SIM-ABLFE game.

The sFULL-SIM-LFE experiment generates $\mathsf{crs}_{\mathsf{LFE}}$, the digest corresponding to $\mathscr{C}$. Then, depending on the value of bit b, the ciphertext is either correctly generated ($b = 0$) or obtained from a simulator ($b = 1$).

The reduction forwards the ciphertext obtained, as well as $\mathsf{CT}_{\mathsf{FHE}}$ and the garbled circuit Γ to $\mathcal{A}_1$. It thus correctly simulate an LFE ciphertext. If $\mathcal{A}_1$ wins the game with advantage ϵ, $\mathcal{B}_1$ wins the sFULL-SIM-ABLFE with a similar advantage[10]. □

Claim (Transition between Game_1 *and* Game_2*).* The advantage of any PPT adversary in distinguishing between Game_1 and Game_2 is:

$$\mathsf{Adv}^{\mathsf{Game}_1 \to \mathsf{Game}_2}_{\mathcal{A}_2}(\lambda) \leq \mathsf{Adv}^{\mathsf{FULL\text{-}SIM\text{-}GS}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) \ .$$

Proof ($\mathsf{Game}_1 \to \mathsf{Game}_2$*).* The simulation security of the garbling scheme we make use of guarantees the existence of a simulator $\mathcal{S}_{\mathsf{GS}}$ that produces a tuple $(\tilde{\Gamma}, \{\tilde{L}^0_i, \tilde{L}^1_i\}_{i\in[t]})$.

We define by $\mathcal{B}_2$ the reduction we build, and let $\mathcal{A}_2$ stand for the adversary against the LFE game. As usual, $\mathcal{B}_2$ starts by sampling and publishing $\mathsf{crs}_{\overline{\mathsf{LFE}}}$, while $\mathcal{A}_2$ provides $(\mathscr{C}, \mathsf{m}^*)$. $\mathcal{B}_2$ concocts the FHE ciphertext. This step is followed by another one, where $\mathcal{B}_2$ plays the role of an adversary against the GS security

[10] Up to a constant loss because of the usage of a two-outcome AB-LFE.

experiment. $\mathcal{B}_2$ provides the GS game with the challenge tuple $(\mathscr{C}_{aux}, \mathscr{C}(\mathsf{m}^*))$, and receives either some correctly generated garbled circuit and labels or a simulated garbled circuit and the simulated labels. Note that $\mathcal{B}_2$ can assemble the AB-LFE ciphertext component.

Directly, if $\mathcal{A}_2$ distinguishes between the two settings, $\mathcal{B}_2$ distinguishes between the two distributions of labels in the GS game. □

Claim (Transition between Game_2 *and* Game_3*).* The advantage of any PPT adversary to distinguish between Game_2 and Game_3 is:

$$\mathsf{Adv}^{\mathsf{Game}_2 \to \mathsf{Game}_3}_{\mathcal{A}_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{FULL\text{-}SIM\text{-}FHE}}_{\mathsf{FHE},\mathcal{B}_3}(\lambda) \ .$$

Proof ($\mathsf{Game}_2 \to \mathsf{Game}_3$*).*

By the semantic security of the underlying FHE scheme, the distribution of ciphertexts $\mathsf{CT}_{\mathsf{FHE}}$ encrypting m is indistinguishable from the distribution of ciphertexts encrypting some constant 0. Therefore, any adversary noticing this transition will distinguish between Game_2 and Game_3. □

We also note that this setting simulates the sFULL-SIM-LFE experiment with $b = 1$. □

A.1 Laconic Parameters

As shown in the motivational part of Section 3, we enhance the size of the LWE modulus:

$$|q| \approx 7\log(n) + 4 \cdot d$$

Furthermore, we can show that our scheme achieves laconic parameters as well, in the sense that they depend on the depth of the supported circuit.

Digest: The digest of our LFE protocol is identical to the AB-LFE counterpart in Definition 5. The latter consists of ℓ matrices, one corresponding to each output bit. Those matrices are computed using the public evaluation algorithm introduced in [GV15]. Therefore, $|\mathsf{digest}_{\mathscr{C}}| \in \ell \cdot \mathsf{poly}(\log(n) + \log(d))$.

Ciphertext: The ciphertext of our LFE scheme consists of three main component. The garbled circuit Γ, the FHE ciphertext $\mathsf{CT}_{\mathsf{FHE}}$, as well as the AB-LFE ciphertext component. The first two components are trivial to analyse: by notation the FHE ciphertext has length t, where t is a polynomial in the input length k. The garbled circuit should perform the FHE decryption (which belongs to NC^1), and therefore, the garbled circuit has a succinct size.
Each of t AB-LFE ciphertexts have succinct size, consisting of a $(k + \ell) \cdot \mathsf{poly}(\log(n) + \log(d))$ number of elements of size $|q|$.

Common Reference String Finally, the common reference string consists of $k + 6$ matrices. Those include the k matrices used to encrypt the input, the 5 matrices representing a state of the branching program, as well as an auxiliary matrix. Therefore, we can conclude that the size of crs depends exclusively on depth d, rather than on the size of $\mathscr{C}$: $|\mathsf{crs}| \in k \ldots \mathsf{poly}(\log(n) + \log(d))$.

B Warm-Up: the LFE for General Circuits in [QWW18]

In this section, we provide a compressed overview of LFE scheme introduced in [QWW18]. Our rationale for doing so are twofold: first, we are going to use some intermediary results described in [QWW18]; second, we can zoom into its internal working in order to have a good point of comparison with the protocol put forth in Section 3. Regarding the protocol itself, four successive steps are required, that we sketched below.

First, the authors show how to achieve an *attribute-based* laconic function evaluation scheme supporting a specific *conditional disclosure functionality*[11] starting from a specific LWE-based ABE construction [BGG+14].

Second, the previous construction is slightly changed to support circuits having multi-bit outputs.

Third, the latter construction is made laconic: the key idea consists in using a laconic oblivious transfer scheme [CDG+17], such that a laconic digest is generated. On Bob's side, a circuit emulating the real LFE encryption procedure is garbled, the labels being LOT-encrypted. The crux point is to garble the circuit that has the crs and the message hardwired, while taking the LFE digest as input.

Finally, in the fourth step, an LFE for general circuits is obtained through the means of a levelled FHE scheme. The techniques used in this part are closely related to the ones presented in [GKP+13].

B.1 Attribute-Based Laconic Function Evaluation

The Basic Construction for AB-LFE. In [QWW18], the authors introduce LFE supporting *Conditional Disclosure Functionality*:

Definition 7 (Conditional Disclosure Functionality). *Let $\mathscr{C} : \{0,1\}^k \to \{0,1\}^\ell$ be a circuit representation of some function f. We define the Conditional Disclosure Functionality as:*

$$\mathsf{CDC}_{\mathscr{C}}(\mathbf{x}, (\mathbf{y}_1, \ldots, \mathbf{y}_\ell)) := (\mathbf{x}, (\overline{\mathbf{y}_1}, \ldots, \overline{\mathbf{y}_\ell})) \quad \textit{where}$$
$$\overline{\mathbf{y}_j} = \begin{cases} \mathbf{y}_j & \textit{if } \mathscr{C}_j(\mathbf{x}) = 0 \ . \\ \perp & \textit{if } \mathscr{C}_j(\mathbf{x}) = 1 \ . \end{cases} \tag{4}$$

and $\mathbf{x} \in \{0,1\}^k$, $\mathbf{y}_j \in \{0,1\}^w$ and $\mathscr{C}_j$ is the j-th output bit of $\mathscr{C}(M)$.

The formal definition of attribute-based laconic function follows.

Definition 8 (Attribute-Based LFE). *An attribute-based laconic function evaluation protocol* AB-LFE *for a class of circuits $\mathcal{C}_\lambda$ that represents a conditional disclosure of secrets* $\mathsf{CDC}_{\mathscr{C}}$ *is similar to an* LFE *protocol for $\mathcal{C}_\lambda$ except for the following changes:*

- $\mathsf{CT} \leftarrow_{\$} \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\mathbf{x}, \mathbf{y}))$*: the randomized procedure encrypts takes as input the digest, but also the plaintext $\mathbf{y}$ and some attribute set $\mathbf{x}$.*

[11] Described below.

– $\mathsf{Dec}(\mathsf{crs}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathsf{CT})$*: Evaluates* $\mathscr{C}$ *over the attributes* $\mathbf{x}$ *and if* $\mathscr{C}(\mathbf{x}) = 1$*, outputs* $\mathbf{y}$*.*

The security definition is essentially similar to the one in Figure 1, up to the change that the adversary issues $(\mathscr{C}, (\mathbf{x}^*, \mathbf{y}^*))$ *in the challenge phase.*

$\underline{\mathsf{sFULL\text{-}SIM\text{-}ABLFE}^{\mathcal{A}}_{\mathsf{AB\text{-}LFE}}(\lambda)}$:	$\underline{\mathsf{FULL\text{-}SIM\text{-}ABLFE}^{\mathcal{A}}_{\mathsf{AB\text{-}LFE}}(\lambda)}$:
$b \leftarrow\!\!\$\ \{0,1\}$	$b \leftarrow\!\!\$\ \{0,1\}$
$((k, d, \mathbf{x}^*, \mathbf{y}^*), \mathscr{C}) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$	$(k, d) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$\mathsf{crs} \leftarrow\!\!\$\ \mathsf{LFE.crsGen}(1^\lambda, k, d)$	$\mathsf{crs} \leftarrow\!\!\$\ \mathsf{LFE.crsGen}(1^\lambda, 1^k, 1^d)$
$\mathsf{digest}_{\mathscr{C}} \leftarrow\!\!\$\ \mathsf{LFE.Compress}(\mathsf{crs}, \mathscr{C})$	$((\mathbf{x}^*, \mathbf{y}^*), \mathscr{C}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs})$
if $b = 0$:	$\mathsf{digest}_{\mathscr{C}} \leftarrow\!\!\$\ \mathsf{LFE.Compress}(\mathsf{crs}, \mathscr{C})$
$\quad \mathsf{CT}^* \leftarrow\!\!\$\ \mathsf{AB\text{-}LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\mathbf{x}^*, \mathbf{y}^*))$	if $b = 0$:
else	$\quad \mathsf{CT}^* \leftarrow\!\!\$\ \mathsf{AB\text{-}LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\mathbf{x}^*, \mathbf{y}^*))$
$\quad \mathsf{CT}^* \leftarrow\!\!\$\ \mathcal{S}(\mathsf{crs}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathbf{y}^*)$	else
$b' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}, \mathsf{CT}^*)$	$\quad \mathsf{CT}^* \leftarrow\!\!\$\ \mathcal{S}(\mathsf{crs}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathbf{y}^*)$
return $b = b'$	$b' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{CT}^*)$
	return $b = b'$

Fig. 2. The selective and adaptive experiments for the case of *attribute-based* LFE. We stress the AB-LFE simulator receives the outcome of $\mathsf{CDC}_{\mathscr{C}}(\alpha)$, that is $\mathbf{y}^*$, and **not** $\mathbf{x}$.

We avoid describing QWW18's *one-bit* of output attribute-base LFE construction since it would be implicit in the multiple-output version. Their attribute-based LFE construction from LWE outputting multiple bits is summarized below:

$\mathsf{crsGen}(1^\lambda, 1^k, 1^d)$**:**
For every input bit $i \in \{1, 2, \ldots, k\}$, a matrix $\mathbf{A}_i$ is sampled:

$$\mathbf{A}_i \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times m} \ .$$

The crs is assigned the following set matrices:

$$\mathsf{crs} \leftarrow \left(\mathbf{A}_1, \ldots, \mathbf{A}_k\right) .$$

$\mathsf{Compress}(\mathsf{crs}, \mathscr{C})$**:**
Let $\mathscr{C} : \{0,1\}^k \to \{0,1\}^\ell$. Output:

$$\mathsf{digest}_{\mathscr{C}} \leftarrow \{\boldsymbol{A}_{\mathscr{C}_j}\}_{j \in [\ell]} = \mathsf{Eval}_{\mathsf{PK}}(\mathscr{C}, \{\boldsymbol{A}_i\}_{i \in [k]}) \ ,$$

where $\mathsf{Eval}_{\mathsf{PK}}$ is defined over multi-bit outputs.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\mathsf{m}, (\mu_1, \ldots, \mu_\ell)))$**:**
Pick $\vec{s} \in \mathbb{Z}_q^n, \vec{e}_i \leftarrow\!\!\$\ \chi^m$ and for each $i \in [k]$ compute:

$$\vec{b}_i \leftarrow \vec{s}^\top \cdot (\boldsymbol{A}_i - \mathsf{m}_i \cdot \boldsymbol{G}) + \vec{e}_i^\top$$

where $\boldsymbol{G}$ is the Gadget matrix. For each $j \in [\ell]$ sample $\boldsymbol{T}_j \leftarrow\!\!\$\ (\boldsymbol{G}^{-1}(\$))^w \in \mathbb{Z}_q^{n\times w}$ and set $\boldsymbol{D}_j \leftarrow \boldsymbol{A}_{\mathscr{C}_j} \cdot \boldsymbol{T}_j$. Then compute $\forall j \in [\ell]$:

$$\beta_j \leftarrow \vec{s}^{\top} \cdot \boldsymbol{D}_j + \overrightarrow{e_j}^{\top} + \mu_j \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q^{1\times w} \ ,$$

where $\overrightarrow{e_j} \leftarrow\!\!\$\ [-B; B]^w$. Output:

$$\mathsf{CT} \leftarrow (\{\vec{b}_i\}_{i\in k}, \{\beta_j\}_{j\in[\ell]}, \mathsf{m}) \ .$$

$\mathsf{Dec}(\mathsf{crs}, \mathscr{C}, \mathsf{digest}_{\mathscr{C}}, \mathsf{CT})$:

Compute

$$\left\{\overrightarrow{b}_{j\ j\in[\ell]} \leftarrow \mathsf{Eval}_{CT}(\mathscr{C}, \{\boldsymbol{A}_i\}_{i\in[k]}, \{\vec{b}_i\}_{i\in[k]}, \{\boldsymbol{T}_j\}_{j\in[\ell]}, \mathsf{m})\right\} .$$

Then, for all $j \in [\ell]$ if $\mathscr{C}_j(\mathsf{m}) = 1$ we set $\mu_j \leftarrow \perp$. Otherwise, set $\mu_j \leftarrow \lfloor(\beta_j - \overrightarrow{b_j} \cdot \boldsymbol{T}_j)/q\rceil$. Finally, return $(\mu_1, \ldots, \mu_\ell)$.

Compressing the Digest. In [QWW18], the authors propose a method to compress the digest through the usage of laconic oblivious transfer. For the sake of self-containess, we reproduce their transform:

Let LOT denote a laconic oblivious transfer scheme (e.g. [CDG+17]). Let GS denote a garbling scheme, and let $\mathsf{AB\text{-}LFE}$ denote an attribute-based laconic function evaluation scheme. One can construct an attribute-based $\mathsf{AB\text{-}LFE}' = (\mathsf{crsGen}', \mathsf{Compress}', \mathsf{Enc}', \mathsf{Dec}')$ with a compressed digest as follows as follows:

$\mathsf{crsGen}'(1^\lambda, 1^k, 1^d)$**:**

Compute $\mathsf{crs}_{\mathsf{LOT}} \leftarrow\!\!\$\ \mathsf{LOT.crsGen}(1^\lambda, 1^k, 1^d)$, $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{AB\text{-}LFE.crsGen}(1^\lambda, 1^k, 1^d)$ and set

$$\mathsf{crs}' \leftarrow (\mathsf{crs}_{\mathsf{LOT}}, \mathsf{crs}) \ .$$

$\mathsf{Compress}'(\mathsf{crs}', \mathscr{C})$**:**

Compute $\mathsf{digest}_{\mathscr{C}} \leftarrow\!\!\$\ \mathsf{AB\text{-}LFE.Compress}(\mathsf{crs}, \mathscr{C})$ and then

$$(\mathsf{digest}', \hat{D}) \leftarrow\!\!\$\ \mathsf{LOT.Compress}(\mathsf{crs}_{\mathsf{LOT}}, \mathsf{digest}_{\mathscr{C}})$$

Return digest'.

$\mathsf{Enc}'(\mathsf{crs}', \mathsf{digest}', (\mathsf{m}, (\mu_1, \ldots, \mu_\ell)))$**:**

Let $E(\cdot)$ be the circuit with $\mathsf{crs}, \mathsf{m}, (\mu_1, \ldots, \mu_\ell)$ and the random coins r hardwired, taking as input $\mathsf{digest}_{\mathscr{C}}$ and outputting $\mathsf{AB\text{-}LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\mathsf{m}, (\mu_1, \ldots, \mu_\ell)))$. The circuit is garbled using GS and $(\Gamma, \{L_i^0, L_i^1\}_{i\in[t]})$ is obtained. Let

$$\mathsf{CT}_i \leftarrow\!\!\$\ \mathsf{LOT.Enc}(\mathsf{crs}_{\mathsf{LOT}}, (i, L_i^0, L_i^1)_{i\in[t]})$$

and return as ciphertext the folowing tuple:

$$\mathsf{CT}' \leftarrow (\Gamma, \{\mathsf{CT}_i\}_{i\in[t]}) \ .$$

$\mathsf{Dec}'(\mathsf{crs}', \mathscr{C}, \mathsf{CT}')$: Compute $\mathsf{digest}_{\mathscr{C}} \leftarrow \mathsf{AB\text{-}LFE.Compress}(\mathsf{crs}, \mathscr{C})$ and then $(\mathsf{digest}', \hat{D}) \leftarrow \mathsf{LOT.Compress}(\mathsf{crs}_{\mathsf{LOT}}, \mathsf{digest}_{\mathscr{C}})$. Decrypt and recover the labels as

$$L_i \leftarrow \mathsf{LOT.Dec}^{\hat{D}}(\mathsf{crs}, \mathsf{CT}_i) \ ,$$

compute $\mathsf{CT} \leftarrow \Gamma.\mathsf{Eval}(\Gamma, \{L_i\}_{i\in[k]})$ and output:

$$\mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, \mathscr{C}, \mathsf{CT}) \ .$$

B.2 LFE for General Circuits via FHE

A transformation relying on FHE is put forward in [QWW18]. We note it follows closely the one introduced in [GKP+13].

Let $\mathcal{C}$ denote a class of circuits of depth d taking k bits as input. Let FHE denote a levelled fully homomorphic encryption scheme supporting circuits of depth $d' = \mathsf{poly}(\lambda, d)$. Let AB-LFE denote a laconic function evaluation scheme. One can construct an LFE supporting circuits in $\mathcal{C}$ as follows:

$\mathsf{crsGen}(1^\lambda, 1^k, 1^d)$:
Return $\mathsf{crs} \leftarrow_\$ \mathsf{AB\text{-}LFE}(1^\lambda, 1^{poly(\lambda,k,d)}, 1^{poly(\lambda,d)})$.

$\mathsf{Compress}(\mathsf{crs}, \mathscr{C})$:
Return $\mathsf{digest}_{\mathscr{C}} \leftarrow_\$ \mathsf{AB\text{-}LFE.Compress}(\mathsf{crs}, \mathsf{FHE.Eval}(\mathscr{C}, \cdot))$

$\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, \mathsf{m})$:
Generate on the fly $(\mathsf{hsk}, \mathsf{hpk}) \leftarrow_\$ \mathsf{FHE.Setup}(1^\lambda, 1^d)$ and compute

$$\overline{\mathsf{m}} \leftarrow_\$ \mathsf{FHE.Enc}(\mathsf{hpk}, \mathsf{m}) \ .$$

Then compute as follows:

$$\begin{aligned}(\Gamma, \{L_i^0, L_i^1\}_{i\in[t]}) &\leftarrow_\$ \Gamma.\mathsf{Garble}(\mathsf{FHE.Dec}(\mathsf{hsk}, \cdot)) \\ \mathsf{CT}_{\mathsf{AB\text{-}LFE}} &\leftarrow_\$ \mathsf{AB\text{-}LFE.Enc}(\mathsf{crs}, \mathsf{digest}_{\mathscr{C}}, (\overline{\mathsf{m}}, \{L_i^0, L_i^1\}_{i\in[t]})) \\ \mathsf{CT} &\leftarrow (\Gamma, \mathsf{CT}_{\mathsf{AB\text{-}LFE}})\end{aligned}$$

$\mathsf{Dec}(\mathsf{crs}, \mathscr{C}, \mathsf{CT})$:
Let $\{L_i\}_{i\in[t]} \leftarrow \mathsf{AB\text{-}LFE.Dec}(\mathsf{crs}, \mathsf{FHE.Eval}(\mathscr{C}, \cdot), \mathsf{CT}_{\mathsf{AB\text{-}LFE}})$.
Return $\Gamma.\mathsf{Eval}(\Gamma, \{L_i\}_{i\in[t]})$.

We state formally the result in [QWW18], which refers to the transformation above.

Lemma 2 (Compiler AB-LFE to LFE for General Circuits [QWW18]). *Assuming the existence of secure garbling schemes and secure fully homomorphic evaluation. Then, any attribute-based laconic function evaluation scheme can be turned into a laconic function evaluation protocol.*