Feasibility and Infeasibility of Adaptively Secure Fully Homomorphic Encryption

Jonathan Katz[*] Aishwarya Thiruvengadam[*] Hong-Sheng Zhou[*][†]

Abstract

Fully homomorphic encryption (FHE) is a form of public-key encryption that enables arbitrary computation over encrypted data. The past few years have seen several realizations of FHE under different assumptions, and FHE has been used as a building block in many cryptographic applications.

Adaptive security for public-key encryption schemes is an important security notion that was proposed by Canetti et al. over 15 years ago. It is intended to ensure security when encryption is used within an interactive protocol, and parties may be *adaptively* corrupted by an adversary during the course of the protocol execution. Due to the extensive applications of FHE to protocol design, it is natural to understand whether adaptively secure FHE is achievable.

In this paper we show two contrasting results in this direction. First, we show that adaptive security is *impossible* for FHE satisfying the (standard) *compactness* requirement. On the other hand, we show a construction of adaptively secure FHE that is not compact, but which does achieve circuit privacy.

Keywords: Fully homomorphic encryption, Adaptive security

1 Introduction

1.1 Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) is a form of public-key encryption that enables a third party (who does not know the associated secret key) to perform computations over encrypted data. That is, given a public key pk and a ciphertext $c = \mathsf{Enc}_{pk}(m)$ that is the encryption of some (unknown) plaintext message m, anyone can compute a ciphertext c' whose decryption is $f(m)$ for any desired function f. The actual definition is even more general (see Section 2.1): given pk and ciphertexts

$$c_1 = \mathsf{Enc}_{pk}(m_1), \ldots, c_\ell = \mathsf{Enc}_{pk}(m_\ell),$$

it is possible to compute an encryption of $f(m_1, \ldots, m_\ell)$.

FHE has several applications. As just one example, FHE can be used to construct simple protocols for secure computation. We restrict ourselves to the two-party setting with honest-but-curious parties. (In this setting two parties with inputs x and y, respectively, wish to compute a function $f(x, y)$ over their inputs without revealing to each other anything more than the result; in the honest-but-curious setting, the parties are again assumed to follow the protocol though privacy of their inputs must still be maintained.) Here, a party with input x can generate a public/private

[*]Department of Computer Science, University of Maryland. Email: {jkatz,aish,hszhou}@cs.umd.edu.

[†]Supported by an NSF CI postdoctoral fellowship.

key pair (pk, sk) for an FHE scheme and send $pk, \mathsf{Enc}_{pk}(x)$ to the other party. This second party can then compute an encryption of the desired result $f(x, y)$ and send the resulting ciphertext back to the first party. Finally, the first party can decrypt this ciphertext to recover $f(x, y)$; it then sends this result to the second party (assuming the second party should also learn the result).

It turns out that FHE with the functionality described above can be realized trivially by the construction in which we simply define $f, \mathsf{Enc}_{pk}(m_1), \ldots, \mathsf{Enc}_{pk}(m_\ell)$ to be a valid encryption of $f(m_1, \ldots, m_\ell)$. This notion, however, does not suffice for most proposed applications of FHE (in particular, it does not suffice for the one above). Thus, some "non-triviality" requirement must be added to the definition of FHE in order to make the notion meaningful. Various such requirements can be considered. We consider two requirements here: (1) *compactness*, which requires that ciphertexts have bounded length, and (2) *circuit privacy*, which requires that the encryption of $f(m_1, \ldots, m_\ell)$ should not reveal f. Note that the trivial scheme described earlier does not satisfy either of these conditions.

Because of its many applications, FHE has long been a "holy grail" in cryptography [RAD78]. It is only in the past few years, however, that candidate FHE schemes have been proposed. The first scheme was constructed by Gentry [Gen09a], and his work inspired a tremendous amount of research showing efficiency improvements to his scheme (e.g., [SS10]), realizations of FHE based on different assumptions (e.g., [vDGHV10, BV11b, BV11a, BGV12, Bra12]), implementations of FHE (e.g., [SV10, GH11]), and new applications of FHE (e.g., [GGP10, CKV10]). We omit further details since they are not directly relevant to our work.

1.2 Adaptive Security

In a separate line of work, the notion of *adaptive security* was proposed for (standard) public-key encryption schemes by Canetti et al. [CFGN96]. Their motivation was to guarantee security when encryption schemes are used to encrypt messages sent during an interactive protocol, and parties running the protocol can be *adaptively* corrupted during the course of the entire protocol execution [BH92] and, in particular, the adversary can determine which parties to corrupt based on what messages have been sent in the protocol thus far. (The adaptive-corruption model stands in contrast to the *static*-corruption model where the attacker is assumed to corrupt parties only *before* the protocol begins.) The adaptive-corruption model is realistic in many environments where protocols may be run, and so it is clearly desirable to achieve this level of security; moreover, it is easy to show protocols secure in the static-corruption model that are demonstrably insecure against adaptive corruptions.

The primary challenge with regard to adaptively secure encryption is that the protocol *simulator* (used to prove security of the protocol) must simulate the ciphertexts being sent by the various parties without knowing the underlying plaintext. At some later point in time, however, the adversary may request to corrupt a party, and the simulator must then simulate for the adversary any secret keys held by that party. (The adversary may also corrupt past *senders* and, if secure erasure is not assumed, the simulator will then have to simulate for the adversary the random tape used by the sender when encrypting. This only makes the problem harder.) These secret keys must be such that they correctly decrypt any ciphertexts previously sent to that party. Most natural public-key encryption schemes will not be suitable here, in particular because a given public key typically has a unique secret key associated with it; this implies that any (correctly generated) ciphertext can be later "opened" to at most one plaintext.

Canetti et al. [CFGN96] show how to construct adaptively secure encryption schemes from general assumptions. Subsequent research (e.g., [Bea97, DN00, JL00, KO04, CHK05, CDSMW09]) has shown more efficient constructions based on specific number-theoretic assumptions, or satisfying weaker (but still meaningful) notions of adaptive security.

1.3 Adaptively Secure FHE?

Because of the various applications of FHE to protocol design, it is natural to ask whether adaptive security can be realized for FHE.

We show two results in this regard. First, we show that adaptive security is *impossible* for FHE schemes satisfying compactness. This result is unconditional, and holds even when only the *receiver* may be corrupted.[1] On the other hand, we show that this notion of adaptive security *is* possible for FHE schemes satisfying circuit privacy. Our results are interesting in their own right, but also show a separation of sorts between two notions of non-triviality (namely, compactness and circuit privacy) that have been considered in the literature.

We remark that the impossibility result of Nielsen [Nie02] does *not* apply to our setting. Nielsen shows that *unbounded* adaptive security is impossible for non-interactive public-key encryption schemes. More precisely, Nielsen shows that if the secret key is t bits long, then it is impossible to achieve adaptive security if $t+1$ bits of plaintext are encrypted. Here, however, we are willing to place an *a priori* upper bound ℓ on the length of the plaintext that will be encrypted and allow the secret key to be arbitrarily long (and, in particular, longer than ℓ). This makes sense when encryption is used to encrypt messages sent within an interactive protocol, where the length of the messages to be encrypted is bounded in advance (and a fresh public key can be generated for each independent protocol execution). And, indeed, when the secret key can be longer than the plaintext, adaptive security is possible for standard public-key encryption schemes.

2 Definitions

Throughout, we let k denote the security parameter.

2.1 Fully Homomorphic Encryption

We begin by formally defining the notion of fully homomorphic encryption. (Although it may appear intuitively obvious how fully homomorphic encryption should be defined, the definition turns out to have a number of subtleties as discussed in the Introduction and further below.)

Definition 2.1 (Fully homomorphic encryption). *Fix a function $\ell = \ell(k)$. An ℓ-*homomorphic encryption scheme *$\mathcal{HE}$ for a class of circuits $\{\mathcal{C}_k\}_{k\in\mathbb{N}}$ consists of four polynomial-time algorithms* Gen*,* Enc*,* Dec*, and* Eval *such that*

- Gen*, the* key-generation algorithm*, is a randomized algorithm that takes the security parameter 1^k as input and outputs a public key pk and secret key sk.*
- Enc*, the* encryption algorithm*, is a randomized algorithm that takes a public key pk and a message $m \in \{0,1\}$ as input, and outputs a ciphertext c.*

[1]An alternate interpretation of our result is that it holds even in a model where secure erasure is possible. In such a model, the sender can erase its randomness immediately after encryption; the receiver, however, cannot erase its secret key until it receives (and decrypts) the ciphertext.

- Dec, *the* decryption algorithm, *is a deterministic algorithm that takes the secret key sk and a ciphertext c as input, and outputs a message $m \in \{0,1\}$.*
- Eval, *the* homomorphic evaluation algorithm, *takes as input a public key pk, a circuit $C \in \mathcal{C}_k$, and a list of ciphertexts*[2] $c_1, \cdots, c_{\ell(k)}$*; it outputs a ciphertext c^*.*

The following correctness properties are required to hold:

1. *For any k, any $m \in \{0,1\}$, and any (pk, sk) output by $\mathsf{Gen}(1^k)$, we have $m = \mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m))$.*
2. *For any k, any $m_1, \ldots, m_\ell$, and any $C \in \mathcal{C}_k$, we have*

$$C(m_1, \ldots, m_\ell) = \mathsf{Dec}_{sk}(\mathsf{Eval}_{pk}(C, \mathsf{Enc}_{pk}(m_1), \ldots, \mathsf{Enc}_{pk}(m_\ell)))$$

(In fact, we can relax the above and require them to hold with all but negligible probability.)

We use the standard notion of security against chosen-plaintext attacks. (Although stronger notions of security could be considered, the question of adaptive security is tangential to these considerations.)

Definition 2.2. *A homomorphic encryption scheme is* secure against chosen-plaintext attacks *(also called* CPA-secure*) if for any polynomial-time adversary $\mathcal{A}$ the following is negligible in k:*

$$|\Pr[\mathcal{A}(pk, \mathsf{Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, \mathsf{Enc}_{pk}(1)) = 1]|,$$

where $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$.

As noted earlier, Definitions 2.1 and 2.2 on their own are not enough to capture a meaningful notion of fully homomorphic encryption. This is because they can be satisfied by a "trivial" construction starting from any CPA-secure (standard) public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$ by defining Enc, Eval, and Dec as follows:

- $\mathsf{Enc}_{pk}(m) = (0, \mathsf{Enc}'_{pk}(m))$.
- $\mathsf{Eval}_{pk}(C, c_1, \ldots, c_\ell)$ outputs $(1, C, c_1, \ldots, c_\ell)$.
- $\mathsf{Dec}_{sk}(c)$ does as follows: if $c = (0, c')$, then output $\mathsf{Dec}'_{sk}(c')$ (i.e., decrypt as in Π). If $c = (1, C, c_1, \ldots, c_\ell)$, then output

$$C(\mathsf{Dec}'_{sk}(c_1), \ldots, \mathsf{Dec}'_{sk}(c_\ell))$$

  (i.e., decrypt and then apply C to the results).

There are various ways one could imagine ruling out trivial schemes like the above. The first approach we consider (following previous work in the literature) is to require that ciphertexts cannot grow arbitrarily large; this is known as *compactness*.

Definition 2.3 (Compactness). *An ℓ-homomorphic encryption scheme $\mathcal{HE}$ for a class of circuits $\{\mathcal{C}_k\}_{k \in \mathbb{N}}$ is* compact *if there exists a polynomial $\alpha = \alpha(k)$ such that ciphertexts output by* Eval *have length at most α. (For this to be non-trivial it should be the case that, for all k, we have $\alpha(k) \leq |C|$ for some $C \in \mathcal{C}_k$.)*

[2] We assume for simplicity that all circuits in $\mathcal{C}_k$ take exactly $\ell = \ell(k)$ input bits.

We say an ℓ-homomorphic encryption scheme is *ℓ-fully homomorphic* if it is homomorphic for all boolean circuits, CPA-secure, and compact.

An alternate non-triviality condition that has been considered is to require that the output of $\mathsf{Eval}_{pk}(C, c_1, \ldots, c_\ell)$ should reveal nothing about C, even to the holder of the secret key sk. This notion is called *circuit privacy*. There are different ways of formalizing such a notion. The definition we use is slightly weaker than some others that have been considered previously, since we allow (an upper bound on) the *size* of C to be revealed. A similar notion was considered in [GHV10].

Definition 2.4 (Circuit privacy). *An ℓ-homomorphic encryption scheme $\mathcal{HE}$ for a class of circuits $\{\mathcal{C}_k\}_{k\in\mathbb{N}}$ is* circuit private *if there exists an efficient simulator $\mathcal{S}$ such that for every (pk, sk) generated by* Gen*, every $C \in \mathcal{C}_k$, and every $m_1, \ldots, m_\ell$, the following two distributions are computationally indistinguishable (even given $pk, sk, C, m_1, \ldots, m_\ell$):*

$$\left\{c_1 \leftarrow \mathsf{Enc}_{pk}(m_1); \ldots, c_\ell \leftarrow \mathsf{Enc}_{pk}(m_\ell) : \left(\mathsf{Eval}_{pk}(C, c_1, \cdots, c_\ell), c_1, \ldots, c_\ell\right)\right\}$$

and

$$\left\{c_1 \leftarrow \mathsf{Enc}_{pk}(m_1); \ldots, c_\ell \leftarrow \mathsf{Enc}_{pk}(m_\ell) : \mathcal{S}(1^k, pk, |C|, C(m_1 \cdots m_\ell), c_1, \cdots, c_\ell)\right\}.$$

We say an ℓ-homomorphic encryption scheme is *circuit-private* homomorphic if it is homomorphic for all boolean circuits, CPA-secure, and circuit private according to Definition 2.4.

2.2 Adaptively Secure Fully Homomorphic Encryption

We consider here a notion of adaptive security for FHE that is weaker than the notion considered by Canetti et al. [CFGN96]. Specifically, we consider only adaptive corruption of the *receiver*. (Alternately, we assume secure erasure and thus the sender can erase the randomness it uses for encryption immediately after encryption is complete.) Here, a simulator is required to produce (a bounded number of) simulated ciphertexts $c_1, \ldots, c_\ell$ that it gives to an adversary; the adversary then outputs messages $m_1, \ldots, m_\ell \in \{0, 1\}$, and the simulator should give the adversary a (single) key sk that "explains" (i.e., decrypts) each ciphertext c_i as m_i.

We stress here that we only require adaptive security where there is an *a priori* upper bound ℓ on the number of encryptions. (This suffices for applications where encryption is used as part of an interactive protocol.) Thus, Nielsen's impossibility result [Nie02] does *not* apply to our setting.

Definition 2.5 (Adaptively secure FHE). *An ℓ-homomorphic encryption scheme $\mathcal{HE}$ = (*Gen, Enc, Dec, Eval*) is* adaptively secure *if there exists a non-uniform, polynomial-time algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all non-uniform, polynomial-time algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have*

$$|\Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(k) = 1] - \Pr[\mathsf{Real}_{\mathcal{A}}(k) = 1]| \leq \mathsf{negl}(k)$$

where:

```
$\underline{\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(k)}$
    $(pk, c_1, \ldots, c_\ell, s) \leftarrow \mathcal{S}_1(1^k)$;
    $(m_1, \ldots, m_\ell, \tau) \leftarrow \mathcal{A}_1(1^k)$;
    $sk \leftarrow \mathcal{S}_2(s, m_1, \ldots, m_\ell)$;
    $b \leftarrow \mathcal{A}_2(\tau, pk, c_1, \ldots, c_\ell, sk)$;
    Return $b$.

$\underline{\mathsf{Real}_{\mathcal{A}}(k)}$
    $(m_1, \ldots, m_\ell, \tau) \leftarrow \mathcal{A}_1(1^k)$;
    $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$;
    $c_1 \leftarrow \mathsf{Enc}_{pk}(m_1); \ldots$;
        $c_\ell \leftarrow \mathsf{Enc}_{pk}(m_\ell)$;
    $b \leftarrow \mathcal{A}_2(\tau, pk, c_1, \ldots, c_\ell, sk)$;
    Return $b$.
```

□

3 Impossibility Result

In this section, we show that adaptively secure ℓ-fully homomorphic encryption is *impossible*. We first give some intuition behind this result. Say adaptively secure FHE were possible. This implies the existence of a simulator as required by Definition 2.5. This then gives an alternate way of computing any function $f : \{0,1\}^\ell \to \{0,1\}$ (described by a circuit C_f), in the following way:

1. Run $\mathcal{S}_1$ to obtain pk, $c_1, \ldots, c_\ell$, and state s.
2. Compute $c' \leftarrow \mathsf{Eval}_{pk}(C_f, c_1, \ldots, c_\ell)$.
3. Given input $x \in \{0,1\}^\ell$, run $\mathcal{S}_2(s, x_1, \ldots, x_\ell)$ to obtain a secret key sk.
4. Compute $\mathsf{Dec}_{sk}(c')$ to obtain $f(x)$.

Note that steps 1 and 2 can be computed in advance of receiving the input x. Thus, we can hard-code s, c', and randomness (if any) for $\mathcal{S}_2$ into a circuit that, upon receiving input $x = (x_1, \ldots, x_\ell)$, computes $sk = \mathcal{S}_2(s, x)$ and then outputs $\mathsf{Dec}_{sk}(c')$. Adaptive security implies that this output must be correct for most inputs x. (More precisely, it guarantees that there *exist* values of s, c', and randomness for $\mathcal{S}_2$ for which the circuit is correct for most inputs x.) But because Dec and $\mathcal{S}_2$ are algorithms of some fixed complexity, and c' is of some bounded size (here is where we use the compactness property), we have some polynomial upper-bound t on the size of the circuit that we get above. Taking f to be a function that cannot be approximated by circuits of size t, but that can be computed in polynomial size $T \gg t$, we arrive at a contradiction.

We now formalize this result.

Theorem 3.1 (Main Result)**.** *Let $\ell = \omega(\log k)$. Then, adaptively secure fully ℓ-homomorphic encryption does not exist.*

Proof. Assume, toward a contradiction, that such a scheme $\mathcal{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ exists. This implies the existence of a non-uniform family of circuits $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ satisfying Definition 2.5. Let $t(k)$ denote an upper bound on the size of the circuit for $\mathcal{S}_2$ plus the size of a circuit computing Dec for any ciphertext c' output by Eval. Using compactness (which says that the size of any such c' is bounded by some fixed polynomial) and the fact that Dec runs in polynomial time, we see that $t(k) = \mathsf{poly}(k)$.

For a given function $f : \{0,1\}^\ell \to \{0,1\}$ that can be computed by a polynomial-size circuit C_f, we will define a circuit $C^*_{s,c',\omega}$. Construction of this circuit proceeds in the following way. First, run $\mathcal{S}_1$ to obtain pk, $c_1, \ldots, c_\ell$, and state s. Then compute $c' \leftarrow \mathsf{Eval}_{pk}(C_f, c_1, \ldots, c_\ell)$. Choose random coins ω for $\mathcal{S}_2$. Then define $C^*_{s,c',\omega}$ as follows:

- On input $x \in \{0,1\}^\ell$, run $\mathcal{S}_2(s, x_1, \ldots, x_\ell)$ (using random coins ω) to obtain sk. Then output $\mathsf{Dec}_{sk}(c')$.

We stress that s, c', and ω are hard-coded into the above circuit. Thus, the size of $C^*_{s,c',\omega}$ is at most $t(k)$.

Given a circuit $C : \{0,1\}^\ell \to \{0,1\}$ and a function $f : \{0,1\}^\ell \to \{0,1\}$, we say that C is an *ϵ-approximation of f* if

$$\Pr_{x \leftarrow \{0,1\}^\ell}[C(x) = f(x)] \geq \epsilon.$$

The theorem follows from the next two lemmas.

Lemma 3.2. *There exist s, c', ω such that the circuit $C^*_{s,c',\omega}$ constructed above is a 3/4-approximation of f.*

Proof. Consider the following non-uniform, polynomial-size adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$: adversary $\mathcal{A}_1$ outputs random $x_1, \ldots, x_\ell \in \{0,1\}$. Then, on input $pk, c_1, \ldots, c_\ell$ and sk, adversary $\mathcal{A}_2$ computes $c' \leftarrow \mathsf{Eval}_{pk}(C_f, c_1, \ldots, c_\ell)$ followed by $y = \mathsf{Dec}_{sk}(c')$. (Non-uniformity is used to hard-wire into $\mathcal{A}_2$ a description of the circuit C_f.) Finally, $\mathcal{A}_2$ outputs 1 if and only if $y = f(x_1, \ldots, x_\ell)$.

Correctness of the FHE scheme implies that in $\mathsf{Real}_\mathcal{A}(k)$ the adversary outputs 1 with all but negligible probability. Adaptive security (Definition 2.5) thus implies that the adversary outputs 1 with all but negligible probability in $\mathsf{Ideal}_\mathcal{A}(k)$. But this means that

$$\Pr_{x,s,c',\omega}[C^*_{s,c',\omega}(x) \neq f(x)] < \mathsf{negl}(k),$$

where $x \in \{0,1\}^\ell$ is chosen uniformly and s, c', ω are generated as in the construction of $C^*_{s,c',\omega}$ described earlier. But this means that there exist s, c', ω for which

$$\Pr_x[C^*_{s,c',\omega}(x) \neq f(x)] < \mathsf{negl}(k),$$

where the probability is now only over the uniform choice of $x \in \{0,1\}^\ell$. This circuit $C^*_{s,c',\omega}$ is thus a 3/4-approximation of f. □

The contradiction is given by the fact that there exist functions f that can be computed by circuits of polynomial size T but cannot be 3/4-approximated by circuits of size t.

Lemma 3.3. *For any $t(k) = \mathsf{poly}(k)$ and $\ell(k) = \omega(\log k)$, there exists a function $f : \{0,1\}^\ell \to \{0,1\}$ that can be computed by a circuit of size $T(k)$, where $2^\ell/\ell > T(k) > \ell$, but which cannot be 3/4-approximated by any circuit of size $t(k)$.*

Proof. A proof follows via suitable modification of the proof of the standard hierarchy theorem for non-uniform computation [AB09]. Pick a random function f, and consider the probability that a fixed circuit C of size S correctly computes f on at least 3/4 of its inputs. Using Chernoff bounds, we can show that this probability is at most $e^{-2^\ell/16}$. Since there are at most $2^{2S\log S + 5S}$ circuits of size S, we have that if $S = 2^{\ell/2}/16$ (and hence $2S \log S + 5S < 2^\ell/16$), then there exists a function that is hard to 3/4-approximate for *all* circuits of size S.

Every function $f : \{0,1\}^\ell \to \{0,1\}$ is computable by a $2^\ell 10\ell$ sized circuit. If we set $\ell = 2.2\ \log\ k$ and let $g : \{0,1\}^k \to \{0,1\}$ be the function that applies f on the first ℓ bits of its input, then g can be computed by a circuit of size $O(k^3)$, but cannot be 3/4-approximated by circuits of size k.

□

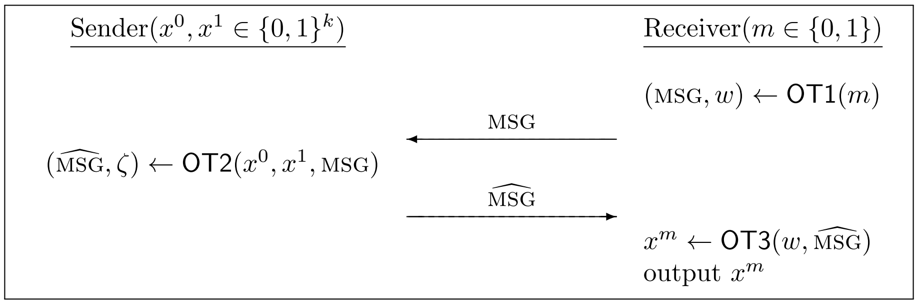
This concludes the proof of the theorem. Thus, for $\ell = \omega(\log k)$, adaptively secure fully ℓ-homomorphic encryption does not exist. □

4 Feasibility Result

In this section, we show that adaptive security is possible for *circuit-private* adaptively secure fully homomorphic encryption (i.e., not requiring the compactness property but satisfying correctness, adaptive security, and circuit privacy). The main idea is similar to the construction in [Gen09b, GHV10, BHHI10] but with adaptively secure building blocks. Specifically, our construction is based on *(i)* a two-move semi-honest oblivious transfer (OT) protocol with receiver adaptive security, in combination with *(ii)* a projective garbling scheme leaking only the circuit size [Yao86, BHR12], and *(iii)* multiple-message, receiver-non-committing public-key encryption (which is a stronger version of single-message receiver-non-committing encryption introduced in [JL00]). Next, we first recall the high-level idea of [Gen09b, GHV10, BHHI10], and then explain how to upgrade the building blocks to achieve our goal.

Oblivious transfer is a cryptographic task carried out between a sender and a receiver, which allows the receiver to obtain one of two values from the sender. The receiver learns only the value he received and nothing about the other value. The sender is oblivious to which value was received by the receiver. Typically, a two-move OT protocol consists of the following steps:

- the receiver based on his input $m \in \{0,1\}$ generates an OT query $(\textsc{msg}, w) \leftarrow \mathsf{OT1}(m)$, where $\textsc{msg}$ is the first-move message, and w is some state used later; the receiver gives $\textsc{msg}$ to the sender;
- then, the sender, based on her input strings (x^0, x^1), generates an OT response $(\widehat{\textsc{msg}}, \zeta) \leftarrow \mathsf{OT2}(x^0, x^1, \textsc{msg})$, where $\widehat{\textsc{msg}}$ is the second-move message, ζ is the internal state used by the sender; the sender gives $\widehat{\textsc{msg}}$ to the receiver;
- finally, based on previously stored state w and the second-move message $\widehat{\textsc{msg}}$ obtained from the sender, the receiver computes the OT output $x^m \leftarrow \mathsf{OT3}(w, \widehat{\textsc{msg}})$.



$\text{Sender}(x^0, x^1 \in \{0,1\}^k)$
$\text{Receiver}(m \in \{0,1\})$
$(\textsc{msg}, w) \leftarrow \mathsf{OT1}(m)$
$\textsc{msg}$
$(\widehat{\textsc{msg}}, \zeta) \leftarrow \mathsf{OT2}(x^0, x^1, \textsc{msg})$
$\widehat{\textsc{msg}}$
$x^m \leftarrow \mathsf{OT3}(w, \widehat{\textsc{msg}})$
output x^m

Figure 1: A 2-move OT protocol.

Interestingly, based on OT, as suggested in [Gen09b, GHV10, BHHI10], we can construct a circuit-private homomorphic scheme, including key generation, encryption, decryption, and evaluation algorithms, as follows: In **key generation**, we can run the key generation of a regular public-key encryption scheme to produce a key-pair (pk, sk). The **encryption** is slightly more complicated, consisting of two steps; to encrypt plaintext $m \in \{0,1\}^\ell$, we first run $\textsc{msg}_i \leftarrow \mathsf{OT1}(m_i; w_i)$ for all $i \in [\ell]$; in order to allow the secret key holder to decrypt, we then include encryption of w_i in the ciphertext based on the public key pk, and let $c_i = (\textsc{msg}_i, \mathsf{Enc}_{pk}(w_i))$ denote the ciphertext for the i-th input bit; the final encryption for m is $c = (c_1, \ldots, c_\ell)$. The **evaluation** algorithm $\mathsf{Eval}(pk, C, c_1, \ldots, c_\ell)$ is based on Yao's garbling technique [Yao86]; we first construct a garbled circuit $\hat{C}$ of circuit C; then we use the OT queries $\{\textsc{msg}_i\}_{i\in[\ell]}$ as well as the encoding of the input wires in $\hat{C}$ to construct OT responses $\{\widehat{\textsc{msg}}_i\}_{i\in[\ell]}$; we finally set $\hat{c}_i := (\widehat{\textsc{msg}}_i, \mathsf{Enc}_{pk}(w_i))$ and output $(\hat{C}, \hat{c}_1, \ldots, \hat{c}_\ell)$ as the ciphertext of the evaluation algorithm. In **decryption** stage, if the received ciphertext is produced by the evaluation algorithm, i.e., in the form of $(\hat{C}, \hat{c}_1, \ldots, \hat{c}_\ell)$, we can first recover w_i by using sk, and obtain the input keys from the OT responses $\widehat{\textsc{msg}}_1, \ldots, \widehat{\textsc{msg}}_\ell$; then we use such input keys to evaluate the garbled circuit $\hat{C}$ to obtain the output. (If the received ciphertext is generated by the encryption algorithm, i.e., $c_i = (\textsc{msg}_i, e_i)$, we can first recover w_i by using sk, and then recover the encrypted bit m_i from $\textsc{msg}_i$ and w_i.)

To achieve adaptive security, we need to update the building blocks with adaptive security. Semi-honest two-move OT protocol will suffice in the above construction; but to achieve adaptive security of the circuit-private homomorphic encryption, we need semi-honest two-move OT but with *adaptive receiver security.* Fortunately, the semi-honest OT protocol in [CLOS02] is sufficient for our goal. We also need to replace the regular public key encryption with a *multi-message receiver non-commiting encryption* (formal definition can be found below). We note that this new notion can be viewed as a strengthened version of receiver non-committing encryption introduced in [JL00]; see [CHK05] for more constructions. Currently, we are not aware as to how such a notion can be achieved in the standard model. Instead, we present a construction in the random oracle (RO) model.

Before describing our construction, we recall the definitions of garbling schemes, semi-honest OT with adaptive receiver security, and then define the multi-message receiver non-committing encryption. Then, we present our construction of a circuit-private homomorphic scheme which achieves correctness, adaptive security, and circuit privacy, but not compactness.

4.1 Building Blocks

4.1.1 Garbling Schemes

Here, we define garbling schemes and introduce the security notion we consider for such schemes in this work. The content here follows the recent work by Bellare, Hoang and Rogaway [BHR12]. For a detailed description of the material presented here, we refer the reader to the original paper [BHR12].

A *garbling scheme* is a five tuple of algorithms $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. A string f, the original function, describes the function $\mathsf{ev}(f, .) : \{0,1\}^\ell \to \{0,1\}^n$ that we want to garble. On input f and a security parameter k, the probabilistic algorithm Gb returns a triple of strings $(F, \mathsf{e}, \mathsf{d}) \leftarrow \mathsf{Gb}(1^k, f)$. String e describes an encoding function, $\mathsf{En}(\mathsf{e}, .)$, that maps an initial input $m \in \{0,1\}^\ell$ to a garbled input $X = \mathsf{En}(\mathsf{e}, m)$. String F describes a garbled function $\mathsf{ev}(F, .)$, that maps each garbled input X to a garbled output $Y = \mathsf{ev}(F, X)$. String d describes a decoding function, $\mathsf{De}(\mathsf{d}, .)$, that maps a garbled output Y to a final output $y = \mathsf{De}(\mathsf{d}, Y)$.

We consider only projective garbling schemes in this work. A *projective garbling scheme* as described in [BHR12] is one where e encodes a list of tokens, one pair for each bit in $m \in \{0,1\}^\ell$. Encoding function $\mathsf{En}(\mathsf{e}, .)$ uses the bits of $m = m_1 \cdots m_\ell$ to select from $\mathsf{e} = X_1^0, X_1^1, \cdots, X_\ell^0, X_\ell^1$ the subvector $X = (X_1^{m_1}, \cdots, X_\ell^{m_\ell})$. A garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is *projective* if for all $f, m, m' \in \{0,1\}^\ell$, $k \in \mathbb{N}$, and $i \in [\ell]$, when $(F, \mathsf{e}, \mathsf{d}) \in [\mathsf{Gb}(1^k, f)]$, $X = \mathsf{En}(\mathsf{e}, m)$ and $X' = \mathsf{En}(\mathsf{e}, m')$, then $X = (X_1 \cdots X_\ell)$ and $X' = (X'_1 \cdots X'_\ell)$ are ℓ vectors, $|X_i| = |X'_i|$, and $X_i = X'_i$ if m and m' have the same ith bit.

For the privacy notion considered, we allow that certain information about the function f can be revealed and this is captured by the *side information function* $\Phi(f)$. Specifically, for this work, the side information function that we allow our garbling scheme to reveal is the size of the circuit.

For the security notion, we describe only the simulation-based notion of privacy in [BHR12]. We note that there are other notions and formulations of security for garbling schemes considered there which could be of interest for later work. We present the definition of the simulation-based security notion of privacy of a garbling scheme.

Definition 4.1. *Consider the following game* $\mathrm{PrvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ *associated with a garbling scheme* $\mathcal{G}$*, side information function* $\Phi(f)$ *and a simulator* $\mathcal{S}$*. The adversary* $\mathcal{A}$ *is run on input* 1^k *and makes exactly one* GARBLE *query. The* GARBLE *procedure is described as follows.*

```
$\underline{\textsc{Garble}(f, m)}$
    $b \leftarrow \{0,1\}$
    $\textit{if } m \notin \{0,1\}^\ell \textit{ return } \bot$
    $\textit{if } b = 1 \textit{ then } (F, \mathsf{e}, \mathsf{d}) \leftarrow \mathsf{Gb}(1^k, f);\quad X \leftarrow \mathsf{En}(\mathsf{e}, m)$
    $\textit{else } y \leftarrow \mathsf{ev}(f, m);\quad (F, X, \mathsf{d}) \leftarrow \mathcal{S}(1^k, y, \Phi(f))$
    $\textit{return } (F, X, \mathsf{d})$
```

The adversary after getting the answer to the query must output a bit b'*. The adversary's advantage is given by:*

$$\mathbf{Adv}_{\mathcal{G}}^{\Phi,\mathcal{S}}(\mathcal{A}, k) = \left|\mathbf{Pr}\left[b' = b\right] - \frac{1}{2}\right|$$

The protocol $\mathcal{G}$ is secure over Φ if for every polynomial-time adversary $\mathcal{A}$ there is a polynomial-time simulator $\mathcal{S}$ such that $\mathbf{Adv}_{\mathcal{G}}^{\Phi,\mathcal{S}}(\mathcal{A}, k)$ is negligible.

4.1.2 Semi-Honest Oblivious Transfer with Adaptive Receiver Security

1-out-of-2 Oblivious Transfer (OT) allows a receiver to obtain exactly one of two messages from a sender where the receiver remains oblivious to the other message, and the sender is oblivious to which value was received. Please refer to Figure 1 for 2-move OT. We next define secure 2-move OT scheme with adaptive receiver security.

Definition 4.2 (Secure 2-move OT with Adaptive Receiver Security)**.** *A k-bit 2-move string oblivious transfer scheme* $\mathcal{OT} = (\mathsf{OT1}, \mathsf{OT2}, \mathsf{OT3}, \mathsf{OT4})$ *is* secure with adaptive receiver security *if the following properties hold:*

Correctness. *The scheme is* correct *if for all* $m \in \{0,1\}$*, and* $x^0, x^1 \in \{0,1\}^k$*, the following hold*

$$\Pr[(\textsc{msg}, w) \leftarrow \mathsf{OT1}(m) : m = \mathsf{OT4}(\textsc{msg}, w)] \leq 1 - \mathsf{negl}(k)$$

$$\Pr[(\textsc{msg}, w) \leftarrow \mathsf{OT1}(m); \widehat{\textsc{msg}} \leftarrow \mathsf{OT2}(x^0, x^1, \textsc{msg})\ :\ x^m = \mathsf{OT3}(w, \widehat{\textsc{msg}})] \leq 1 - \mathsf{negl}(k)$$

Adaptive Receiver Security. *The scheme satisfies* adaptive receiver security *if there exists a non-uniform, polynomial-time algorithm* $\mathcal{S}^{\mathrm{recv}} = (\mathcal{S}_1^{\mathrm{recv}}, \mathcal{S}_2^{\mathrm{recv}})$ *such that for all non-uniform, polynomial-time algorithms* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *we have*

$$|\Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^{\mathrm{recv}}}(k) = 1] - \Pr[\mathsf{Real}_{\mathcal{A}}(k) = 1]| \leq \mathsf{negl}(k)$$

where:

```
$\underline{\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^{\mathrm{recv}}}(k)}$
    $(m, \tau) \leftarrow \mathcal{A}_1(1^k);$
    $(\textsc{msg}, s) \leftarrow \mathcal{S}_1^{\mathrm{recv}}(1^k);$
    $w \leftarrow \mathcal{S}_2^{\mathrm{recv}}(s, m);$
    $b \leftarrow \mathcal{A}_2(\tau, w, \textsc{msg});$
    $\textit{Return } b.$
```

```
$\underline{\mathsf{Real}_{\mathcal{A}}(k)}$
    $(m, \tau) \leftarrow \mathcal{A}_1(1^k);$
    $(\textsc{msg}, w) \leftarrow \mathsf{OT1}(m);$
    $b \leftarrow \mathcal{A}_2(\tau, w, \textsc{msg});$
    $\textit{Return } b.$
```

Sender Security. *The scheme satisfies* sender security *if there exists a non-uniform, polynomial-time algorithm* $\mathcal{S}^{\text{send}}$ *such that for all non-uniform, polynomial-time algorithms* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *we have*

$$|\Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^{\text{send}}}(k) = 1] - \Pr[\mathsf{Real}_{\mathcal{A}}(k) = 1]| \leq \mathsf{negl}(k)$$

where:

```
$\underline{\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^{\text{send}}}(k)}$
    $(x^0, x^1, m, \tau) \leftarrow \mathcal{A}_1(1^k)$;
    $(\text{MSG}, w) \leftarrow \mathsf{OT1}(m)$;
    $\widehat{\text{MSG}} \leftarrow \mathcal{S}^{\text{send}}(1^k, \text{MSG}, x^m)$;
    $b \leftarrow \mathcal{A}_2(\tau, \text{MSG}, w, \widehat{\text{MSG}})$;
    Return $b$.

$\underline{\mathsf{Real}_{\mathcal{A}}(k)}$
    $(x^0, x^1, m, \tau) \leftarrow \mathcal{A}_1(1^k)$;
    $(\text{MSG}, w) \leftarrow \mathsf{OT1}(m)$;
    $\widehat{\text{MSG}} \leftarrow \mathsf{OT2}(x^0, x^1, \text{MSG})$;
    $b \leftarrow \mathcal{A}_2(\tau, \text{MSG}, w, \widehat{\text{MSG}})$;
    Return $b$.
```

We note that adaptively secure semi-honest OT in [CLOS02] satisfies the above properties.

4.1.3 Multi-Message, Receiver-Non-Committing Public-Key Encryption

Receiver non-committing encryption (RNCE) was introduced in [JL00] and further studied in [CHK05]. Here, we strengthen their notion to deal with multiple messages with an a priori bound $\alpha = \mathsf{poly}(k)$ on the number of messages; we call this α-message RNCE, and formally define it below.

- The randomized key-generation algorithm `gen` takes as input the security parameter and outputs a key-pair. This is denoted by: $(\text{pk}, \text{sk}) \leftarrow \texttt{gen}(1^k)$. The public key pk defines the message space $\mathcal{M}$.
- The randomized encryption algorithm `enc` takes a public key pk and a message $m \in \mathcal{M}$. It returns a ciphertext $c \leftarrow \texttt{enc}_{\text{pk}}(m)$.
- The decryption algorithm `dec` takes as input a secret key sk and a ciphertext c, and returns a message $m \leftarrow \texttt{dec}_{\text{sk}}(c)$, where $m \in \mathcal{M} \cup \perp$.
- The randomized key-faking algorithm $\widetilde{\texttt{gen}}$ takes as input the security parameter and outputs a public key as well as some auxiliary information. This is denoted by: $(\text{pk}, z) \leftarrow \widetilde{\texttt{gen}}(1^k)$.
- The fake encryption algorithm $\widetilde{\texttt{enc}}$ takes as input a tuple (pk, z) as output by $\widetilde{\texttt{gen}}$, and outputs a tuple of fake ciphertexts and some auxiliary information: $(c_1, \ldots, c_\alpha, z') \leftarrow \widetilde{\texttt{enc}}(\text{pk}, z)$.
- The reveal algorithm $\widetilde{\texttt{rev}}$ takes as input a tuple $(c_1, \ldots, c_\alpha, z')$ as output by $\widetilde{\texttt{enc}}$, and a tuple of messages $m_1, \ldots, m_\alpha \in \mathcal{M}$. It outputs a fake secret key $\text{sk} \leftarrow \widetilde{\texttt{rev}}(z', c_1, \ldots, c_\alpha, m_1, \ldots, m_\alpha)$. (Intuitively, sk is a valid-looking secret key for which c_i decrypts to m_i for all $i \in [\alpha]$.)

Definition 4.3 (α-Message Receiver Non-Committing Security)**.** *We say that* $(\texttt{gen}, \texttt{enc}, \texttt{dec})$ *is a secure receiver non-committing encryption scheme for bounded* $\alpha = \mathsf{poly}(k)$*, if there exist non-uniform, polynomial-time algorithms* $\mathcal{S} = (\widetilde{\texttt{gen}}, \widetilde{\texttt{enc}}, \widetilde{\texttt{rev}})$ *such that for all non-uniform, polynomial-time algorithms* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *we have*

$$|\Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(k) = 1] - \Pr[\mathsf{Real}_{\mathcal{A}}(k) = 1]| \leq \mathsf{negl}(k)$$

where:

```
$\underline{\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}(k)}$
    $(m_1, \ldots, m_\alpha, \tau) \leftarrow \mathcal{A}_1(1^k)$;
    $(\mathrm{pk}, z) \leftarrow \widetilde{\mathtt{gen}}(1^k)$;
    $(c_1, \ldots, c_\alpha, z') \leftarrow \widetilde{\mathtt{enc}}(\mathrm{pk}, z)$;
    $\mathrm{sk} \leftarrow \widetilde{\mathtt{rev}}(z', c_1, \ldots, c_\alpha, m_1, \ldots, m_\alpha)$;
    $b \leftarrow \mathcal{A}_2(\tau, \mathrm{pk}, c_1, \ldots, c_\alpha, m_1, \ldots, m_\alpha)$;
    Return $b$.

$\underline{\mathsf{Real}_{\mathcal{A}}(k)}$
    $(m_1, \ldots, m_\alpha, \tau) \leftarrow \mathcal{A}_1(1^k)$;
    $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathtt{gen}(1^k)$;
    $c_1 \leftarrow \mathtt{enc}_{\mathrm{pk}}(m_1); \ldots;$
      $c_\alpha \leftarrow \mathtt{enc}_{\mathrm{pk}}(m_\alpha)$;
    $b \leftarrow \mathcal{A}_2(\tau, \mathrm{pk}, c_1, \ldots, c_\alpha, m_1, \ldots, m_\alpha)$;
    Return $b$.
```

It is not clear how to construct a scheme to satisfy the above multiple message receiver non-committing property in the standard model (even assuming that honest entity is allowed to securely erase unnecessary internal state). Following the idea of [BR93, Nie02], here we give a random oracle based construction $(\mathtt{gen}, \mathtt{enc}, \mathtt{dec})$ which satisfies α-message receiver non-committing security defined above.

- The randomized key-generation algorithm: $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathtt{gen}(1^k)$:

  Generate trapdoor permutation G with inverse G^{-1}, as well as a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^k$; set $\mathrm{pk} := (G, H)$ and $\mathrm{sk} := G^{-1}$.

- The randomized encryption algorithm $c \leftarrow \mathtt{enc}_{\mathrm{pk}}(m)$:

  Upon receiving $m \in \{0,1\}^k$, choose r at random from the domain of G and compute $c^1 \leftarrow G(r)$ and $c^2 \leftarrow H(r) \oplus m$. Set $c := (c^1, c^2)$.

- The decryption algorithm $m \leftarrow \mathtt{dec}_{\mathrm{sk}}(c)$:

  Upon receiving c, parse $c = (c^1, c^2)$; then compute $r \leftarrow G^{-1}(c^1)$, $m \leftarrow H(r) \oplus c^2$.

- The randomized key-faking algorithm $(\mathrm{pk}, z) \leftarrow \widetilde{\mathtt{gen}}(1^k)$:

  Generate trapdoor permutation G with inverse G^{-1}; let $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be the random oracle; set $\mathrm{pk} := (G, H)$ and $z := G^{-1}$.

- The fake encryption algorithm $(c_1, \ldots, c_\alpha, z') \leftarrow \widetilde{\mathtt{enc}}(\mathrm{pk}, z)$:

  For all $i \in [\alpha]$, choose r_i at random from the domain of G, compute $c_i^1 \leftarrow G(r_i)$, and choose $c_i^2 \leftarrow \{0,1\}^k$ at random, set $c_i := (c_i^1, c_i^2)$. Set $z' := (\mathrm{pk}, z, \{r_i\}_{i \in [\alpha]})$.

- The reveal algorithm $\mathrm{sk} \leftarrow \widetilde{\mathtt{rev}}(z', c_1, \ldots, c_\alpha, m_1, \ldots, m_\alpha)$:

  Parse $z' = (\mathrm{pk}, z, \{r_i\}_{i \in [\alpha]})$, where $\mathrm{pk} := (G, H)$ and $z := G^{-1}$. Parse $c_i = (c_i^1, c_i^2)$, and set $H(r_i) := c_i^2 \oplus m_i$. Set $\mathrm{sk} := G^{-1}$.

4.2 Adaptively Secure Circuit-Private ℓ-Homomorphic Encryption

Our construction is based on (i) projective garbling scheme $(\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$, leaking only the circuit size [Yao86, LP09, BHR12], (ii) receiver adaptively secure semi-honest OT protocol $(\mathsf{OT1}, \mathsf{OT2}, \mathsf{OT3}, \mathsf{OT4})$, and (iii) ℓ-message receiver non-committing public-key encryption $(\mathtt{gen}, \mathtt{enc}, \mathtt{dec})$. Next we present the construction of a circuit-private ℓ-homomorphic encryption $\mathcal{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$.

- The key-generation algorithm $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$:

  Compute $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathtt{gen}(1^k)$, and set $pk := \mathrm{pk}$ and $sk := \mathrm{sk}$.

- The encryption algorithm $c \leftarrow \mathsf{Enc}_{pk}(m)$:

  Upon input $m \in \{0,1\}$, compute $(\textsc{msg}, w) \leftarrow \mathsf{OT1}(m)$, and then compute $e \leftarrow \mathtt{enc}_{\mathrm{pk}}(w)$. Set $c := (\textsc{msg}, e)$.

- The decryption algorithm $m \leftarrow \mathsf{Dec}_{sk}(c)$:

  – Upon input ciphertext c, parse c. If it parses into $(\textsc{msg}, e)$, then compute $w \leftarrow \mathtt{dec}_{\mathrm{sk}}(e)$, and compute $m \leftarrow \mathsf{OT4}(\textsc{msg}, w)$.

  – Otherwise, if it parses into $(\hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$, then for all $i \in [\ell]$, further parse $\hat{c}_i$ into $(\widehat{\textsc{msg}}_i, e_i)$, compute $w_i \leftarrow \mathtt{dec}_{\mathrm{sk}}(e_i)$, $X_i^{m_i} \leftarrow \mathsf{OT3}(w_i, \widehat{\textsc{msg}}_i)$. This gives the garbled input $X = (X_1^{m_1}, \ldots, X_\ell^{m_\ell})$. Now, output $\mathsf{De}(\mathsf{d}, \mathsf{Ev}(\hat{C}, X))$.

- The evaluation algorithm $c^* \leftarrow \mathsf{Eval}(pk, C, c_1, \ldots, c_\ell)$:

  – Using the projective garbling scheme, on input C, let $\mathsf{Gb}(1^k, C) \rightarrow (\hat{C}, \mathsf{e}, \mathsf{d})$. Parse the encoding function represented by string e as $(X_1^0, X_1^1, \ldots, X_\ell^0, X_\ell^1)$.

  – Parse c_i into $(\textsc{msg}_i, e_i)$ for $i \in [\ell]$. Then compute $\widehat{\textsc{msg}}_i \leftarrow \mathsf{OT2}(X_i^0, X_i^1, \textsc{msg}_i)$, and set $\hat{c}_i := (\widehat{\textsc{msg}}_i, e_i)$, for all $i \in [\ell]$.

  – Finally, set $c^* := (\hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$.

Theorem 4.4. *Construction $\mathcal{HE}$ presented above is an adaptively secure circuit-private ℓ-homomorphic encryption.*

Proof. We show below that the construction $\mathcal{HE}$ is a secure ℓ-homomorphic encryption that satisfies correctness, circuit privacy, and adaptive security.

Correctness. It is easy to verify the correctness. To compute $\mathsf{Dec}_{sk}(c)$ for $c = \mathsf{Enc}_{pk}(m)$ where $c = (\textsc{msg}, e)$ and $e = \mathtt{enc}_{\mathrm{pk}}(w)$, we first compute $w' \leftarrow \mathtt{dec}_{\mathrm{sk}}(e)$, and then compute $m' \leftarrow \mathsf{OT4}(\textsc{msg}, w')$. Given the fact that $(\mathtt{gen}, \mathtt{enc}, \mathtt{dec})$ is correct, it holds that $w = \mathtt{dec}_{\mathrm{sk}}(\mathtt{enc}_{\mathrm{pk}}(w))$, i.e., $w = w'$; furthermore, given the fact that the OT scheme is correct, it holds that $m = \mathsf{OT4}(\mathsf{OT1}(m))$, i.e., $m = m'$. Therefore, we have $\mathsf{Dec}_{sk}(c) = m$ for $c = \mathsf{Enc}_{pk}(m)$.

To compute $\mathsf{Dec}_{sk}(c)$ for $c = \mathsf{Eval}(pk, C, c_1, \ldots, c_\ell)$, where $c = (\hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$, parse $\hat{c}_i$ as $\hat{c}_i = (\widehat{\textsc{msg}}_i, e_i)$ for all $i \in [\ell]$. We first compute $w_i' \leftarrow \mathtt{dec}_{\mathrm{sk}}(e_i)$, $\hat{X}_i \leftarrow \mathsf{OT3}(w_i', \widehat{\textsc{msg}}_i)$. Then we use the input key strings $X = (\hat{X}_1, \ldots, \hat{X}_\ell)$ to evaluate the garbled circuit $\hat{C}$ as $\mathsf{Ev}(\hat{C}, X)$ and obtain $C(\hat{m}_1, \ldots, \hat{m}_\ell)$. Given the fact that $(\mathtt{gen}, \mathtt{enc}, \mathtt{dec})$ is correct, we have $w_i = w_i'$; furthermore, given the fact that the OT is correct, $\hat{X}_i = X_i^{m_i}$; also, given the fact that the garbling scheme is correct, we have $C(m_1, \ldots, m_\ell) = \mathsf{De}(\mathsf{d}, \mathsf{Ev}(\hat{C}, X))$. Therefore, we have $\mathsf{Dec}_{sk}(c) = m$ for $c = \mathsf{Eval}(pk, C, c_1, \ldots, c_\ell)$ and $m = C(m_1, \ldots, m_\ell)$.

Circuit privacy. The property of circuit privacy follows from the security of the garbling scheme and the sender security of the OT. By the security of the garbling scheme, we have that there exists a simulator $\mathcal{S}^{\mathrm{G}}$ on input the security parameter, output of the function and the side information function Φ outputs (F, X, d) except with negligible probability. As mentioned before, $\Phi(C) = |C|$ in our construction.

Let us construct a simulator $\mathcal{S}$ as follows:

- Upon receiving $(1^k, pk, |C|, C(m_1, \ldots, m_\ell), c_1, \ldots, c_\ell)$ where $c_i = \mathsf{Enc}_{pk}(m_i)$ for $i \in [\ell]$, the simulator $\mathcal{S}$ runs the simulator $\mathcal{S}^{\mathrm{G}}$ of the garbling scheme to obtain $(F, X, \mathsf{d}) \leftarrow \mathcal{S}^{\mathrm{G}}(1^k, C(m_1, \ldots, m_\ell), |C|)$.
- Set $\hat{C} = F$. Parse X as $(\hat{X}_1, \ldots, \hat{X}_\ell)$.
- To compute the ciphertext $\hat{c}_i$, parse c_i into $(\textsc{msg}_i, e_i)$ as in the construction. Then compute $\widehat{\textsc{msg}}_i \leftarrow \mathcal{S}^{\text{send}}(1^k, \textsc{msg}_i, \hat{X}_i)$ using the OT-simulator for sender security, and set $\hat{c}_i := (\widehat{\textsc{msg}}_i, e_i)$, for all $i \in [\ell]$.
- Finally, set $c^* := (\hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$.

Next, we develop a sequence of hybrids to show that Definition 2.4 is satisfied.

Hybrid $\mathbf{H}_0$: As in the real scheme, we run the evaluation algorithm Eval to compute c^*, i.e, $c^* \leftarrow \mathsf{Eval}(pk, C, c_1, \ldots, c_\ell)$ where $c^* = (\hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$. Concretely, we use the projective garbling scheme, on input C, compute $(\hat{C}, \mathsf{e}, \mathsf{d}) \leftarrow \mathsf{Gb}(1^k, C)$; then we parse the encoding function represented by string e as $(X_1^0, X_1^1, \ldots, X_\ell^0, X_\ell^1)$, and parse c_i into $(\textsc{msg}_i, e_i)$ for $i \in [\ell]$; after that, we compute $\widehat{\textsc{msg}}_i \leftarrow \mathsf{OT2}(X_i^0, X_i^1, \textsc{msg}_i)$, and set $\hat{c}_i := (\widehat{\textsc{msg}}_i, e_i)$, for all $i \in [\ell]$; finally, set $c^* := (\hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$. The adversary $\mathcal{A}$ is given c^* as well as the input m, circuit C and the secret key sk.

Hybrid $\mathbf{H}_{1,j}$: For $j \in [0 \ldots \ell]$, the hybrid $\mathbf{H}_{1,j}$ is the same as the Hybrid $\mathbf{H}_0$ except the following:

For all $i \in [j]$, parse c_i into $(\textsc{msg}_i, e_i)$. Then compute $\widehat{\textsc{msg}}_i \leftarrow \mathcal{S}^{\text{send}}(1^k, \textsc{msg}_i, X_i^{m_i})$, using the OT simulator for sender security and set $\hat{c}_i := (\widehat{\textsc{msg}}_i, e_i)$.

We argue that for $j \in [0, \ldots, \ell-1]$, the hybrids $\mathbf{H}_{1,j}$ and $\mathbf{H}_{1,j+1}$ are computationally indistinguishable under the assumption that the OT satisfies sender security. If there is an adversary $\mathcal{A}$ who can distinguish between the two hybrids with non-negligible probability, then we can construct an adversary $\mathcal{B}$ who breaks the sender security of the OT as follows.

The adversary $\mathcal{B}$ acts as follows:

- run $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, i.e., run $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{gen}(1^k)$, and set $pk := \mathrm{pk}$, $sk := \mathrm{sk}$.
- Choose $m = (m_1, \ldots, m_\ell)$. For all $i \in [1, \ldots, \ell]$, compute $c_i \leftarrow \mathsf{Enc}_{pk}(m_i)$, i.e., compute $(\textsc{msg}_i, w_i) \leftarrow \mathsf{OT1}(m_i)$, and $e_i \leftarrow \mathsf{enc}_{\mathrm{pk}}(w_i)$, and set $c_i := (\textsc{msg}_i, e_i)$;
- Using the projective garbling scheme on a circuit C, let $\mathsf{Gb}(1^k, C) \to (\hat{C}, \mathsf{e}, \mathsf{d})$. Parse the encoding function represented by string e as $(X_1^0, X_1^1, \ldots, X_\ell^0, X_\ell^1)$;
- for all $i \in [1, \ldots, j-1]$, parse c_i into $(\textsc{msg}_i, e_i)$. Then compute $\widehat{\textsc{msg}}_i \leftarrow \mathsf{OT2}(X_i^0, X_i^1, \textsc{msg}_i)$, and set $\hat{c}_i := (\widehat{\textsc{msg}}_i, e_i)$;
- for $i = j$, output (m_i, X_i^0, X_i^1). Then parse c_i into $(\textsc{msg}_i, e_i)$ and obtain $\widehat{\textsc{msg}}_i$, where it is either generated by $\mathcal{S}^{\text{send}}$, i.e., $\widehat{\textsc{msg}}_i \leftarrow \mathcal{S}^{\text{send}}(1^k, \textsc{msg}_i, X_i^{m_i})$, or generated by the OT scheme honestly, i.e., $\widehat{\textsc{msg}}_i \leftarrow \mathsf{OT2}(X_i^0, X_i^1, \textsc{msg}_i)$;
- for all $i \in [j+1, \ell]$, parse c_i into $(\textsc{msg}_i, e_i)$. Then compute $\widehat{\textsc{msg}}_i \leftarrow \mathcal{S}^{\text{send}}(1^k, \textsc{msg}_i, X_i^{m_i})$;
- for all $i \in [\ell]$, set $\hat{c}_i := (\widehat{\textsc{msg}}_i, e_i)$;
- Return $(m, sk, pk, C, \hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$ to the internally simulated $\mathcal{A}$.

When $i = j$, if $\widehat{\text{MSG}}_i$ obtained by $\mathcal{B}$ is generated by $\mathcal{S}^{\text{send}}$, then $\mathcal{A}$ interacts with Hybrid $\mathbf{H}_{1,j+1}$. Otherwise, if $\widehat{\text{MSG}}_i$ is generated by the OT scheme honestly, then $\mathcal{A}$ interacts with Hybrid $\mathbf{H}_{1,j}$. Based on the assumption that $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, we can conclude that $\mathcal{B}$ can distinguish Ideal from Real as in Definition 4.2 for sender security. This contradicts our assumption that the OT is sender-secure. Therefore, Hybrid $\mathbf{H}_{1,j+1}$ and Hybrid $\mathbf{H}_{1,j}$ are indistinguishable.

Note that $\mathbf{H}_0$ is identical to $\mathbf{H}_{1,0}$. Since Hybrids $\mathbf{H}_{1,j+1}$ and $\mathbf{H}_{1,j}$ are indistinguishable as argued above, we also have that Hybrid $\mathbf{H}_0$ is computationally indistinguishable from Hybrid $\mathbf{H}_{1,\ell}$.

Hybrid $\mathbf{H}_2$: This is the same as Hybrid $\mathbf{H}_{1,\ell}$ except the following:

We run the simulator $\mathcal{S}^{\text{G}}$ for the projective garbling scheme to obtain the garbled circuit, input and the decoding function, i.e., $(F, X, \mathsf{d}) \leftarrow \mathcal{S}^{\text{G}}(1^k, C(m_1, \ldots, m_\ell), |C|)$. Parse X as $(\hat{X}_1, \ldots, \hat{X}_\ell)$, and set $\hat{C} = F$.

We note that this is exactly the output produced by the simulator $\mathcal{S}$ for circuit privacy.

The hybrids $\mathbf{H}_{1,\ell}$ and $\mathbf{H}_2$ are indistinguishable under the assumption that $\mathcal{G}$ is a secure garbling scheme. If there is an adversary $\mathcal{A}$ who can distinguish between the two hybrids with non-negligible probability, then we can construct an adversary $\mathcal{B}$ who breaks the security of the garbling scheme as defined in Definition 4.1.

Consider an adversary $\mathcal{B}$ who acts as follows:

- run $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$;
- Choose $m = (m_1, \ldots, m_\ell)$. For all $i \in [1, \ldots, \ell]$, compute $c_i \leftarrow \mathsf{Enc}_{pk}(m_i)$;
- Choose a circuit C. Make a GARBLE query on (C, m) to obtain the challenge (F, X, d). Set $\hat{C} = F$.
- Parse X as $(\hat{X}_1, \ldots, \hat{X}_\ell)$. For all $i \in [\ell]$, parse c_i into (MSG_i, e_i), and compute $\widehat{\text{MSG}}_i \leftarrow \mathcal{S}^{\text{send}}(1^k, \text{MSG}_i, \hat{X}_i)$; Set $\hat{c}_i = (\widehat{\text{MSG}}_i, e_i)$;
- Return $(m, sk, pk, C, \hat{C}, \mathsf{d}, \hat{c}_1, \ldots, \hat{c}_\ell)$ to the internally simulated $\mathcal{A}$.

When $\mathcal{B}$'s challenge (F, X, d) is generated honestly, then the internally simulated $\mathcal{A}$ interacts with $\mathbf{H}_{1,\ell}$. On the other hand, when $\mathcal{B}$'s challenge is generated by $\mathcal{S}^{\text{G}}$, the simulated $\mathcal{A}$ interacts with $\mathbf{H}_2$. Based on the assumption that $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, we can conclude that $\mathcal{B}$ can gain a non-negligible advantage as in Definition 4.1. However, this is a contradiction to our assumption that $\mathcal{G}$ is a secure garbling scheme. Therefore, Hybrid $\mathbf{H}_{1,\ell}$ and Hybrid $\mathbf{H}_2$ are indistinguishable.

Thus, we have that the hybrids $\mathbf{H}_0$ and $\mathbf{H}_2$ are indistinguishable which implies that the scheme satisfies the circuit privacy requirement as defined in Definition 2.4.

ADAPTIVE SECURITY. Next we give the proof idea for proving the adaptive security as defined in Definition 2.5. We construct the simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ as follows. The simulator is based on the algorithms $(\widetilde{\mathtt{gen}}, \widetilde{\mathtt{enc}}, \widetilde{\mathtt{rev}})$ of the RNCE scheme, and the simulator $(\mathcal{S}_1^{\text{recv}}, \mathcal{S}_2^{\text{recv}})$ of the OT scheme.

- $(pk, c_1, \dots, c_\ell, s) \leftarrow \mathcal{S}_1(1^k)$:

  Compute $(\mathrm{pk}, z) \leftarrow \widetilde{\mathtt{gen}}(1^k)$, and set $pk := \mathrm{pk}$. Compute $(e_1, \dots, e_\ell, z') \leftarrow \widetilde{\mathtt{enc}}(\mathrm{pk}, z)$.

  For all $i \in [\ell]$, compute $(\textsc{msg}_i, \gamma_i) \leftarrow \mathcal{S}_1^{\mathrm{recv}}(1^k)$, and set $c_i := (\textsc{msg}_i, e_i)$.

  Store all information into state s.

- $sk \leftarrow \mathcal{S}_2(s, m_1, \dots, m_\ell)$:

  Upon obtaining $(m_1, \dots, m_\ell)$, recover $\{\gamma_i\}_{i\in[\ell]}$ from the state s.

  For all $i \in [\ell]$, compute $w_i \leftarrow \mathcal{S}_2^{\mathrm{recv}}(\textsc{msg}_i, \gamma_i, m_i)$.

  Compute $\mathrm{sk} \leftarrow \widetilde{\mathtt{rev}}(z', \{e_i, w_i\}_{i\in[\ell]})$, and set $sk := \mathrm{sk}$.

Next, we develop a sequence of hybrids to show that the real experiment defined in Definition 2.5 is indistinguishable from the ideal experiment.

Hybrid $\mathbf{H}_0$: This is the real experiment.

When the adversary $\mathcal{A}$ outputs $(m_1, \dots, m_\ell)$, compute $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathtt{gen}(1^k)$, and set $pk := \mathrm{pk}$, $sk := \mathrm{sk}$. Then for all $i \in [\ell]$, do the following: compute $(\textsc{msg}_i, w_i) \leftarrow \mathsf{OT1}(m_i)$, $e_i \leftarrow \mathtt{enc}_{\mathrm{pk}}(w_i)$, and set $c_i := (\textsc{msg}_i, e_i)$. Return $(pk, c_1, \dots, c_\ell, sk)$ to the adversary.

Hybrid $\mathbf{H}_{1,j}$: Here $j \in [0, \dots, \ell]$. The hybrid is the same as the Hybrid $\mathbf{H}_0$ except the following:

For all $i \in [j]$, compute $(\textsc{msg}_i, \gamma_i) \leftarrow \mathcal{S}_1^{\mathrm{recv}}(1^k)$. Then for all $i \in [j]$, compute $w_i \leftarrow \mathcal{S}_2^{\mathrm{recv}}(\textsc{msg}_i, \gamma_i, m_i)$.

We argue that for $j \in [0, \dots, \ell-1]$, the hybrids $\mathbf{H}_{1,j}$ and $\mathbf{H}_{1,j+1}$ are computationally indistinguishable under the assumption that the OT satisfies adaptive receiver security. Assume there is an adversary $\mathcal{A}$ who can distinguish between the two hybrids. For all $\mathcal{S}^{\mathrm{recv}}$, we next show how to construct $\mathcal{B}$ to distinguish between the Real experiment from the Ideal experiment as in Definition 4.2.

$\mathcal{B}$ internally simulates $\mathcal{A}$ and receives $(m_1, \dots, m_\ell)$ from it. Then $\mathcal{B}$ carries out the following:

- run $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathtt{gen}(1^k)$, and set $pk := \mathrm{pk}$, $sk := \mathrm{sk}$;
- for all $i \in [1, \dots, j-1]$, compute $(\textsc{msg}_i, w_i) \leftarrow \mathsf{OT1}(m_i)$, $e_i \leftarrow \mathtt{enc}_{\mathrm{pk}}(w_i)$;
- for $i = j$, obtain the pair $(\textsc{msg}_i, w_i)$, where the pair is generated by either $\mathcal{S}^{\mathrm{recv}} = (\mathcal{S}_1^{\mathrm{recv}}, \mathcal{S}_2^{\mathrm{recv}})$, i.e., $(\textsc{msg}_i, \gamma_i) \leftarrow \mathcal{S}_1^{\mathrm{recv}}(1^k)$ and $w_i \leftarrow \mathcal{S}_2^{\mathrm{recv}}(\textsc{msg}_i, \gamma_i, m_i)$, or generated by the OT scheme honestly, i.e., $(\textsc{msg}_i, w_i) \leftarrow \mathsf{OT1}(m_i)$;
- for all $i \in [j+1, \ell]$, run $(\textsc{msg}_i, \gamma_i) \leftarrow \mathcal{S}_1^{\mathrm{recv}}(1^k)$ and $w_i \leftarrow \mathcal{S}_2^{\mathrm{recv}}(\textsc{msg}_i, \gamma_i, m_i)$;
- for all $i \in [\ell]$, set $c_i := (\textsc{msg}_i, e_i)$;
- Return $(pk, c_1, \dots, c_\ell, sk)$ to the internally simulated $\mathcal{A}$.

Note that when $i = j$, if the pair $(\textsc{msg}_i, w_i)$ that $\mathcal{B}$ obtains are generated by $\mathcal{S}^{\mathrm{recv}} = (\mathcal{S}_1^{\mathrm{recv}}, \mathcal{S}_2^{\mathrm{recv}})$, then $\mathcal{A}$ interacts with Hybrid $\mathbf{H}_{1,j+1}$, while if the pair $(\textsc{msg}_i, w_i)$ that $\mathcal{B}$ obtains are generated by the OT scheme honestly, $\mathcal{A}$ interacts with Hybrid $\mathbf{H}_{1,j}$. Based on the assumption that $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, we can conclude that $\mathcal{B}$ can distinguish Ideal from Real as in Definition 4.2 for defining adaptive receiver security. This leads to a contradiction to our assumption that the OT has adaptive receiver security. Therefore, Hybrid $\mathbf{H}_{1,j+1}$ and Hybrid $\mathbf{H}_{1,j}$ are indistinguishable.

Note that $\mathbf{H}_0$ is identical to $\mathbf{H}_{1,0}$. Since Hybrid $\mathbf{H}_{1,j+1}$ and Hybrid $\mathbf{H}_{1,j}$ are indistinguishable, as argued above, we also have that $\mathbf{H}_0$ are computationally indistinguishable from $\mathbf{H}_{1,\ell}$.

Hybrid $\mathbf{H}_2$: The hybrid is the same as Hybrid $\mathbf{H}_{1,\ell}$ except the following:

Compute $(\mathrm{pk}, z) \leftarrow \widetilde{\mathtt{gen}}(1^k)$, and set $pk := \mathrm{pk}$. Compute $(e_1, \ldots, e_\ell, z') \leftarrow \widetilde{\mathtt{enc}}(\mathrm{pk}, z)$. Compute $\mathrm{sk} \leftarrow \widetilde{\mathtt{rev}}(z', \{e_i, w_i\}_{i\in[\ell]})$, and set $sk := \mathrm{sk}$. We note that this is exactly the ideal experiment.

We argue that $\mathbf{H}_{1,\ell}$ and $\mathbf{H}_2$ are computationally indistinguishable under the ℓ-message RNC security. Assume there is an adversary $\mathcal{A}$ who can distinguish between the two hybrids. We next show how to construct a machine $\mathcal{B}$ to win the ℓ-message RNC security game.

$\mathcal{B}$ receives pk, and internally runs $\mathcal{A}$. Upon receiving $(m_1, \ldots, m_\ell)$ from $\mathcal{A}$, $\mathcal{B}$ computes $(\textsc{msg}_i, w_i) \leftarrow \mathsf{OT1}(m_i)$ for all $i \in [\ell]$, and outputs $(w_1, \ldots, w_i)$ to its challenger. Upon receiving from its challenger $(e_1, \ldots, e_\ell, \mathrm{sk})$, $\mathcal{B}$ defines $c_i := (\textsc{msg}_i, e_i)$ for all $i \in [\ell]$, and returns $(pk, c_1, \ldots, c_\ell, sk)$ to $\mathcal{A}$. $\mathcal{B}$ returns $\mathcal{A}$'s output as its own output.

When $\mathcal{B}$'s received tuple $(\mathrm{pk}, e_1, \ldots, e_\ell, \mathrm{sk})$ is generated by $(\mathtt{gen}, \mathtt{enc}, \mathtt{dec})$ honestly, then the internally simulated $\mathcal{A}$ interacts with $\mathbf{H}_{1,\ell}$. On the other hand, when $\mathcal{B}$'s received tuple $(\mathrm{pk}, e_1, \ldots, e_\ell, \mathrm{sk})$ is generated by $(\widetilde{\mathtt{gen}}, \widetilde{\mathtt{enc}}, \widetilde{\mathtt{rev}})$, i.e, $(\mathrm{pk}, z) \leftarrow \widetilde{\mathtt{gen}}(1^k)$, $(e_1, \ldots, e_\ell, z') \leftarrow \widetilde{\mathtt{enc}}(\mathrm{pk}, z)$, and $\mathrm{sk} \leftarrow \widetilde{\mathtt{rev}}(z', \{e_i, w_i\}_{i\in[\ell]})$, the simulated $\mathcal{A}$ interacts with $\mathbf{H}_2$. Based on the assumption that $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, we can conclude that $\mathcal{B}$ can distinguish Ideal from Real as in Definition 4.3. This leads to a contradiction. Therefore, Hybrids $\mathbf{H}_{1,\ell}$ and $\mathbf{H}_2$ are indistinguishable.

Based on the above argument we have $\mathbf{H}_0$ and $\mathbf{H}_{1,\ell}$ are indistinguishable, and $\mathbf{H}_{1,\ell}$ and $\mathbf{H}_2$ are indistinguishable. Therefore $\mathbf{H}_0$ and $\mathbf{H}_2$ are indistinguishable. Note that Hybrid $\mathbf{H}_2$ is exactly the ideal experiment, and $\mathbf{H}_0$ is the real experiment. We now have that the ideal and the real experiments are indistinguishable. This implies that the scheme satisfies adaptive security. $\square$

References

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[Bea97] Donald Beaver. Plug and play encryption. In Burton S. Kaliski Jr., editor, *Advances in Cryptology — Crypto '97*, volume 1294 of *LNCS*, pages 75–89. Springer, 1997.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science (ITCS)*, pages 309–325. ACM, 2012.

[BH92] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *Advances in Cryptology — Eurocrypt '92*, volume 658 of *LNCS*, pages 307–323. Springer, 1992.

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *Advances in Cryptology — Eurocrypt 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, 2010.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Garbling schemes. Cryptology ePrint Archive, Report 2012/265, 2012. `http://eprint.iacr.org/`.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conf. on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, pages 868–886, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS 2011*, pages 97–106. IEEE, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology — Crypto 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, 2011.

[CDSMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *Advances in Cryptology — Asiacrypt 2009*, volume 5912 of *LNCS*, pages 287–302. Springer, 2009.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648. ACM Press, May 1996.

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In Joe Kilian, editor, *2nd Theory of Cryptography Conference — TCC 2005*, volume 3378 of *LNCS*, pages 150–168. Springer, February 2005.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Advances in Cryptology — Crypto 2010*, volume 6223 of *LNCS*, pages 483–501. Springer, 2010.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503. ACM Press, 2002.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *Advances in Cryptology — Crypto 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, 2000.

[Gen09a] C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178. ACM Press, 2009.

[Gen09b] Craig Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology — Crypto 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, 2010.

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology — Eurocrypt 2011*, volume 6632 of *LNCS*, pages 129–148. Springer, 2011.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In *Advances in Cryptology — Crypto 2010*, volume 6223 of *LNCS*, pages 155–172. Springer, 2010.

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, 2000.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *Advances in Cryptology — Crypto 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, 2004.

[LP09] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[Nie02] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology — Crypto 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, 2002.

[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *Advances in Cryptology — Asiacrypt 2010*, volume 6477 of *LNCS*, pages 377–394. Springer, December 2010.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *13th Intl. Conference on Theory and Practice of Public Key Cryptography — PKC 2010*, volume 6056 of *LNCS*, pages 420–443. Springer, 2010.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology — Eurocrypt 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, 2010.

[Yao86] A. C.-C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167. IEEE, 1986.