CS6501 Topics in Cryptography, Spring 2015

Project: Functional Controllable Homomorphic Encryption

Group Members: Ameer Mohammed (am8zv), Soheil Nemati (sn8fb)

---

1 Introduction

In this project, we define and propose a construction of a primitive called *functional controllable homomorphic encryption* (FCHE) that shares some properties of functional encryption (FE) and controllable homomorphic encryption (CHE). Ideally, such a hybrid encryption scheme should possess two primary properties. The first property is derived from CHE, which states that, given some encryption $c = \text{Enc}(m)$ of a message m and a privately-generated token for some function g, one can invoke an evaluation procedure on c using the token for g to get $\text{Enc}(g(m))$. The second property is derived from FE and builds upon any CHE ciphertext. That is, given any ciphertext $c = \text{Enc}(g(m))$ (where g the identity function if this is a fresh CHE encryption), one can decrypt this ciphertext using a secret key (token) representing some function f to finally get $f(g(m))$.

The building blocks for our primitive consist of controllable homomorphic encryption and a variant of weak obfuscation called token-based obfuscation [5] where, given an obfuscated circuit, an evaluator can only evaluate on an input if it has a secretly generated token for this input. For the purposes of exposition, our resulting scheme is a restrictive form of FCHE, where the decryptor must submit its ciphertext to the key generation entity *in addition* to the function f that it wants to evaluate on the encrypted message m. We suspect that a true (non-restrictive) FCHE can be constructed using FE and stronger forms of obfuscation in a fashion much similar to the techniques described in [4], but we have yet to find a way that is also provably secure.

2 Motivation

To motivate the study of FCHE, we devised the following scenario in which such a primitive might be useful. Consider an organization having several clients whose information is stored by a trusted party (such as a system administrator) on a remote server. This server might have been leased by the organization from an external agency in order to outsource data storage and computation and, therefore, would not be completely trusted. Our goal in this system is twofold: first, we would like the server to perform computations on the data while maintaining its secrecy, and second, we would like the users to only be able to decipher the part of the encrypted data that belongs to them.

The first requirement can be satisfied using any FHE scheme by sending this data in an encrypted format using a fully homomorphic encryption scheme to allow computation on the data by the server. However, we would not want the server to manipulate the data in an undesirable way, be it intentional or otherwise, so we would like to restrict the range of functions that the server can use in the evaluation. To achieve this, we use *controllable homomorphic encryption*, introduced in [3], so that an evaluator for function g is required to have a secret evaluation token EK_g, which is generated by the trusted party, to compute this function over the encrypted input.

The second requirement, which is selective decryption, can be achieved using functional encryption. However, we cannot naively add a standalone FE scheme to encrypt the message in

conjunction with a homomorphic encryption scheme since the ciphertext of an FE scheme cannot be normally manipulated by a HE scheme. Thus we should somehow endow an existing functional encryption with homomorphic evaluation capabilities (or alternatively, start with a HE scheme and modify it to allow for selective decryption).

Consequently, we define FCHE as a way to solve the above two requirements by encapsulating in one scheme both the ability for the server to homomorphically evaluate a function g on a ciphertext $c = \mathrm{Enc}(m)$ (given an evaluation token EK_g for g) and then later for the user to decrypt it using a secret token SK_f to get $f(g(m))$.

Remark 2.1. A natural generalization of FCHE that one may also consider is the notion of functional *fully* homomorphic encryption. However, we note that such a construction may be infeasible to achieve under the combined security definitions of FE and FHE. That is also why we resort to relaxing the homomorphic properties in order to realize a more reasonable security definition.

3 Definitions

We list here some common definitions of primitives that we use throughout this work. Those familiar with these primitives can skip this section. For the definition of controllable homomorphic encryption (CHE), see Section 4.1. For the definition of functional controllable homomorphic encryption (FCHE), see Section 5.

Definition 3.1 (Functional Encryption [2,6])**.** For security parameter κ, a functional encryption scheme FE for a family of deterministic functions $\mathcal{F}$ is a set of four algorithms defined as follows:

- $\mathsf{FE}.\mathrm{Setup}(1^\kappa)$: a PPT algorithm that outputs the master public key MPK and the master secret key MSK. MPK is used for any and all encryptions of messages and MSK is used to derive decryption keys for functions.
- $\mathsf{FE}.\mathrm{Keygen}(\mathsf{MSK}, f)$: a PPT algorithm that uses the master secret key MSK to derive a secret key SK_f for function $f \in \mathcal{F}$.
- $\mathsf{FE}.\mathrm{Enc}(\mathsf{MPK}, x)$: a PPT algorithm that encrypts message x using MPK and outputs ciphertext c.
- $\mathsf{FE}.\mathrm{Dec}(\mathsf{SK}_f, c)$: a deterministic algorithm that uses secret key SK_f on ciphertext c to get message x'

Definition 3.2 (FE Correctness)**.** A functional encryption scheme FE is said to be correct if for all $f \in \mathcal{F}$,

$$\Pr\left[\begin{array}{l}(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE}.\mathrm{Setup}(1^\kappa);\\ \mathsf{SK}_f \leftarrow \mathsf{FE}.\mathrm{Keygen}(\mathsf{MSK}, f);\\ \mathsf{FE}.\mathrm{Dec}(\mathsf{SK}_f, \mathsf{FE}.\mathrm{Enc}(\mathsf{MPK}, x)) = f(x)\end{array}\right] = 1 - \mathrm{negl}(\kappa)$$

We define two notions of security for functional encryption: indistinguishability security where the adversary asks for q keys and simulation-based security where the adversary asks for 1 key only.

Definition 3.3 (FE q-IND-CPA Security)**.** A functional encryption scheme FE is said to be q-IND-CPA secure if for all PPT adversaries A:

$$\Pr[\text{IND-CPA}_A^{\mathsf{FE}}(1^\kappa) = 1] \leq \frac{1}{2} + \text{negl}(\kappa)$$

where $\text{IND-CPA}_A^{\mathsf{FE}}$ is shown in Figure 1, and for each of the q queries that A sends to the $\mathsf{FE}.\text{Keygen}$ oracle in Step 2 and 6, it must hold that $f(x_0) = f(x_1)$. An adaptive adversary is one that has oracle access to the $\mathsf{FE}.\text{Keygen}$ oracle in both Steps 2 and 6 of Figure 1, whereas a non-adaptive adversary might have only one or the other.

Experiment $\text{IND-CPA}_A^{\mathsf{FE}}(1^\kappa)$:
1. $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE}.\text{Setup}(1^\kappa)$
2. $(x_0, x_1) \leftarrow A^{\mathsf{FE}.\text{Keygen}(\mathsf{MSK},.)}(\mathsf{MPK})$
3. $b \xleftarrow{\$} \{0,1\}$
4. $c_b \leftarrow \mathsf{FE}.\text{Enc}(\mathsf{MPK}, x_b)$
5. $b' \leftarrow A^{\mathsf{FE}.\text{Keygen}(\mathsf{MSK},.)}(\mathsf{MPK}, c_b)$
6. Output $(b = b')$

Figure 1: The $\text{IND-CPA}_A^{\text{FE}}$ Experiment

Definition 3.4 (FE 1-SIM-Security)**.** A functional encryption scheme FE is said to be 1-SIM-secure if for all PPT adversaries A there exists a simulator such that:

$$\{\mathsf{Real}_A^{\mathsf{FE}}(1^\kappa)\}_\kappa \approx_c \{\mathsf{Ideal}_A^{\mathsf{FE}}(1^\kappa)\}_\kappa$$

where the experiments $\mathsf{Real}_A^{\mathsf{FE}}(1^\kappa)$ and $\mathsf{Ideal}_A^{\mathsf{FE}}(1^\kappa)$ are shown in Figure 2.

Experiment $\mathsf{Real}_A^{\mathsf{FE}}(1^\kappa)$:
1. $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE}.\text{Setup}(1^\kappa)$
2. $f \leftarrow A(\mathsf{MPK})$
3. $\mathsf{SK}_f \leftarrow \mathsf{FE}.\text{Keygen}(\mathsf{MSK}, f)$
4. $x \leftarrow A(\mathsf{MPK}, \mathsf{SK}_f)$
5. $c \leftarrow \mathsf{FE}.\text{Enc}(\mathsf{MPK}, x)$
6. $\alpha \leftarrow A(\mathsf{MPK}, c_b)$
7. Output (α, c)

Experiment $\mathsf{Ideal}_A^{\mathsf{FE}}(1^\kappa)$:
1. $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE}.\text{Setup}(1^\kappa)$
2. $f \leftarrow A(\mathsf{MPK})$
3. $\mathsf{SK}_f \leftarrow \mathsf{FE}.\text{Keygen}(\mathsf{MSK}, f)$
4. $x \leftarrow A(\mathsf{MPK}, \mathsf{SK}_f)$
5. $c_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{MPK}, \mathsf{SK}_f, f, f(x), 1^{|x|})$
6. $\alpha \leftarrow A(\mathsf{MPK}, c_{\mathsf{Sim}})$
7. Output (α, c)

Figure 2: The simulation-based experiments

Note that it was shown in [1] that q-SIM-security for FE schemes is impossible to achieve since, in that case, the ciphertext size would grow linearly with q. Also, *adaptive* 1-SIM-security was shown to be impossible to achieve by [2].

Functional encryption generalizes the concept of attribute-based encryption (ABE). In particular, the class of functions supported by ABE are those that output the message if a predicate on the encrypted message's attribute is satisfied and output $\perp$ otherwise.

Definition 3.5 (Garbling Scheme)**.** A *garbling* scheme Gb for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ consists of three PPT algorithms ($\mathsf{Gb}.\,\mathrm{Garble}, \mathsf{Gb}.\,\mathrm{Enc}, \mathsf{Gb}.\,\mathrm{Eval}$) defined as follows:

- $\mathsf{Gb}.\,\mathrm{Garble}(1^\kappa, C)$: a PPT algorithm takes as input the security parameter κ and circuit $C \in \mathcal{C}_n$ then outputs a garbled circuit $\widetilde{C}$ and secret encoding key sk
- $\mathsf{Gb}.\,\mathrm{Enc}(\mathsf{sk}, x)$: Takes as input the secret encoding key and the input x then outputs an encoding c
- $\mathsf{Gb}.\,\mathrm{Eval}(\widetilde{C}, c)$: Takes a garbled circuit and corresponding encoding of the input and outputs $\widetilde{C}(c)$

Definition 3.6 (Garbling Scheme Correctness)**.** A garbling scheme is said to be correct if the following is satisfied for all n, circuits $C \in \mathcal{C}_n$ and inputs $x \in \{0,1\}^n$:

$$\Pr[\mathsf{Gb}.\,\mathrm{Eval}(\widetilde{C}, \mathsf{Gb}.\,\mathrm{Enc}(\mathsf{sk}, x)) = C(x)] = 1 - \mathrm{negl}(\kappa)$$

where $(\mathsf{sk}, \widetilde{C}) \leftarrow \mathsf{Gb}.\,\mathrm{Garble}(C)$ and the probability is over the randomness of the three algorithms

Definition 3.7 (Garbling Scheme Security)**.** A garbling scheme is said to be input- and circuit-private if, for all PPT adversaries A, there exists a simulator $\mathsf{Sim}_{\mathsf{Gb}}$ such that the following holds:

$$|\Pr[A(C, x, \widetilde{C}, c) = 1] - \Pr[A(C, x, \widetilde{C}_{\mathsf{Sim}}, c_{\mathsf{Sim}} = 1] = \mathrm{negl}(\kappa)$$

where C, x are the circuit and input chosen by A, $\widetilde{C}$ is output by $\mathsf{Gb}.\,\mathrm{Garble}(1^\kappa, C)$, $c \leftarrow \mathsf{Gb}.\,\mathrm{Enc}(\mathsf{sk}, x)$, and $(\widetilde{C}_{\mathsf{Sim}}, c_{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_{\mathsf{Gb}}(1^\kappa, C(x), 1^{|C|}, 1^{|x|})$. Intuitively, this means that the adversary should learn nothing more than what it would learn in the simulated world since $\mathsf{Sim}_{\mathsf{Gb}}$ knows neither C nor x.

4 Summary of Related Work

In this section, we summarize the techniques and results of the papers from which we initiate our study of FCHE. The first paper introduces the concept of controllable homomorphic encryption, and the second is a construction of functional encryption from LWE-based primitives among other applications, one of which we will make use of.

4.1 Controllable Homomorphic Encryption

We discuss here the main results of [3]. In the paper, the authors propose a new primitive which is called controllable homomorphic encryption (CHE) that borrows some features from both fully homomorphic encryption and functional encryption. By using CHE, one can (similarly to a FHE) homomorphically evaluate a ciphertext $Ct = \mathrm{Enc}(m)$ and a circuit C therefore obtaining $\mathrm{Enc}(C(m))$ but only if (similarly to functional encryption) a token for C has been received from the owner of the secret key.

4.1.1 Definition

The main intuition behind their construction is that they change the circuit C to new circuit C' which is similar to the encryption of the circuit C. To understand this change, we need to see what is the definition of CHE.

Definition 4.1. A *controllable homomorphic encryption* scheme (CHES) is a five-tuple $\mathsf{CHE} = (\mathsf{CHE.Setup}, \mathsf{CHE.Keygen}, \mathsf{CHE.Enc}, \mathsf{CHE.HEval}, \mathsf{CHE.Dec})$ of efficient algorithms with the following syntax:

- $\mathsf{CHE.Setup}(1^\kappa, 1^n)$: On input the security parameter κ and length parameter n, output public and master secret keys $(\mathsf{MPK}, \mathsf{MSK})$.
- $\mathsf{CHE.Keygen}(\mathsf{MSK}, C)$: On input the master secret key MSK and the description of an $n-bit$ input and $n-bit$ output circuit C, outputs token EK_C for circuit C.
- $\mathsf{CHE.Enc}(\mathsf{MPK}, x)$: On input the public key MPK with and plaintext $x \in \{0,1\}^n$, outputs a ciphertext Ct.
- $\mathsf{CHE.HEval}(\mathsf{MPK}, Ct, \mathsf{EK}_C)$: On input the public key MPK, a ciphertext Ct for plaintext $x \in \{0,1\}^n$ and a token EK_C for circuit C, outputs a string Ct of size independent of C.
- $\mathsf{CHE.Dec}(\mathsf{MSK}, Ct)$: On input the master secret key MSK and a string Ct, outputs a string $x \in \{0,1\}^n$.

In the definition we encrypt once by using $\mathsf{CHE.Enc}$ then one can use two functions $\mathsf{CHE.HEval}$ and $\mathsf{CHE.Dec}$ to either evaluate or decrypt the ciphertext. The first idea for the construction we can encrypt twice during the encryption phase ($\mathsf{CHE.Enc}$). Therefore, when we want to use $\mathsf{CHE.HEval}$ and $\mathsf{CHE.Dec}$ we decrypt both ciphertexts at the same time. The other idea is that when one wants to evaluate a circuit C, we send the token of the other circuit C' which is $C'(x) = \mathrm{Enc}(C(x))$. Now we want to formalize the notation of security. The authors defined a security game which is equivalent to non-malleable chosen-plaintext attack NM-CPA security.

Definition 4.2 (CHE NM-CPA Security)**.** A CHE is a NM-CPA secure if for every PPT adversary A and all polynomially bounded $n = n(\kappa)$ we have that the following two ensembles are indistinguishable:

$$\{\text{CHE-NMCPA-Game}_{0,A}^{\mathsf{CHE}}(\kappa, n)\} \text{ and } \{\text{CHE-NMCPA-Game}_{1,A}^{\mathsf{CHE}}(\kappa, n)\}$$

where the $\text{CHE-NMCPA-Game}_{b,A}^{\mathsf{CHE}}(\kappa, n)$ experiment is defined in the following sequence of steps that the challenger takes:

1. **Setup**: Computes $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{CHE.Setup}(1^\kappa, 1^n)$ then runs A on input MPK.
2. **Token Query**: Replies to a token query for a circuit C by returning $\mathsf{EK}_C \leftarrow \mathsf{CHE.Keygen}(\mathsf{MSK}, C)$.
3. i^{th} **Encryption Query**: Replies to an encryption query (x_0^i, x_1^i) where $x_0^i = x_1^i$ and $|x_0^i| = |x_1^i|$, by returning $Ct \leftarrow \mathsf{CHE.Enc}(\mathsf{MPK}, x_b^i)$.
4. **Output of the Game**. Let (j, Ct, C) be A's output. If all the following conditions hold:
   - A did not issue a token query for circuit C that coincides with C on x_b^j .
   - $C(x_0^j) = C(x_1^j)$.
   - $\mathsf{CHE.Dec}(\mathsf{MSK}, Ct) = C(x_b^j)$.
   - $Ct^\star$ is not a ciphertext obtained as a reply to an encryption query.

   then the output is $C(x_b^j)$. Otherwise the output of the game is $\perp$.

4.1.2 CHE Construction

In the paper the authors show that we can get CHE from FE. First they defined a new circuit which is a modified version of the original circuit. In the description of CHE, we let FE =(FE. Setup, FE. Enc, FE. Keygen, FE. Eval) be an IND-CPA secure non-rerandomizable tag based FE4C. For a CHE with n-bit plaintexts and security parameter κ, we use an FE for plaintexts of length n and an auxiliary message of length $\kappa + 2$. In addition, we let $F = \{F(.,.)\}$ be a pseudorandom family of functions $F(.,.)$ (the first argument is the seed), and let $\mathsf{PKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme. Finally, we define the new circuit C' as follows:

$$C_{s,\mathsf{FE.pk}}(x, r, t, \mathsf{sk}) = \begin{cases} \mathsf{FE.}\,\text{Enc}(\mathsf{FE.pk}, (C(x), 0^\kappa, \perp, 0^\kappa); F(r, C)) & t = 0 \\ \text{Dec}(\mathsf{sk}, s) & t = 1 \\ \perp & t = \perp \end{cases}$$

Given the elements described above, we can now describe the algorithm which they propose for CHE.

Construction 4.3. CHE Algorithm:

- Algorithm $\mathsf{CHE.}\,\text{Setup}(1^\kappa, 1^n)$
  1. Run algorithm $\mathsf{FE.}\,\text{Setup}$ on input $(1^\kappa, 1^n + 2\kappa + 2)$ and obtain $(\mathsf{FE.pk}, \mathsf{FE.MSK})$
  2. Run algorithm Setup on input 1^κ and obtain $(\mathsf{pk}, \mathsf{sk})$
  3. Set $\mathsf{MPK} = \mathsf{FE.pk}$ and $\mathsf{MSK} = (\mathsf{FE.pk}, \mathsf{FE.MSK}, \mathsf{pk})$
  4. Return $(\mathsf{MPK}, \mathsf{MSK})$
- Algorithm $\mathsf{CHE.}\,\text{Keygen}(\mathsf{MSK}, C)$
  1. Set $s = \text{Enc}(\mathsf{pk}, \mathsf{FE.}\,\text{Enc}(\mathsf{FE.pk}, (0^n, 0^\kappa, \perp, 0^\kappa)))$
  2. Set $\mathsf{EK}_C = \mathsf{FE.}\,\text{Keygen}(\mathsf{FE.MSK}, C_s)$
  3. Return EK_C
- Algorithm $\mathsf{CHE.}\,\text{Enc}(\mathsf{MPK}, x)$
  1. Randomly select tag $r \in \{0,1\}^\kappa$
  2. Run algorithm Setup on input 1^κ and obtain $(\mathsf{pk}, \mathsf{sk})$
  3. Set $Ct = \mathsf{FE.}\,\text{Enc}(\mathsf{FE.pk}, (x, r, 0, \mathsf{sk}))$
  4. Return Ct
- Algorithm $\mathsf{CHE.}\,\text{HEval}(\mathsf{MPK}, Ct, \mathsf{EK}_C)$: Outputs $\mathsf{FE.}\,\text{Eval}(\mathsf{FE.pk}, Ct, \mathsf{EK}_C)$
- Algorithm $\mathsf{CHE.}\,\text{Dec}(\mathsf{MSK}, Ct)$
  1. Let $\mathsf{EK}_{ID} = \mathsf{FE.}\,\text{Keygen}(\mathsf{FE.MSK}, ID)$ where the circuit ID is defined in the following way: $ID(x_1, x_2, x_3, x_4) = x_1$
  2. Return $\mathsf{FE.}\,\text{Eval}(\mathsf{FE.pk}, Ct, \mathsf{EK}_{ID})$

4.1.3 Security

In this section we sketch the proof of security of which there are five hybrids. First we are going to define the first hybrid H_0^β then we mention the difference between the hybrids.
Hybrid H_0^β.

1. Setup: Set $(\mathsf{pk}, \mathsf{sk}) \leftarrow \text{Setup}(1^\kappa)$ and $(\mathsf{FE.pk}, \mathsf{FE.sk}) \leftarrow \mathsf{FE.}\,\text{Setup}(1^\kappa, 1^{n+2\kappa+2})$. Then, randomly pick $r, r' \leftarrow \{0,1\}^\kappa$ and run A on input $\mathsf{CHE.pk} = \mathsf{FE.pk}$.

2. Encryption query: When A issues an encryption query for messages (x_0, x_1), return $Ct = \mathsf{FE.}\,\text{Enc}(\mathsf{FE.pk}, (x_\beta, r, 0, \mathsf{sk}))$.

3. Token query: When A issues a token query for circuit C, proceed as follows: Pick random $z \in \{0,1\}^\kappa$ and set $s = \mathsf{FE.}\,\text{Enc}(\mathsf{FE.pk}, (0^n, 0^\kappa, \perp, 0^\kappa); z)$. Set $s = \text{Enc}(\mathsf{pk}, s)$ and $\mathsf{EK}_C = \mathsf{FE.}\,\text{Keygen}(\mathsf{FE.sk}, C_s)$ and return EK_C.

Other hybrids are the same as hybrid 0 and we thus only show the difference between them in the following table.

Hybrid	Plaintext for Ct	s in Tok
H_0^β	$(M_\beta, r, 0, sk)$	$FE.Enc(FE.Pk, (0^n, 0^\kappa, \perp, 0^\kappa); z)$
H_1^β	$(M_\beta, r, 0, sk)$	$FE.Enc(FE.Pk, (0^n, 0^\kappa, \perp, 0^\kappa); F(r', C))$
H_2^β	$(M_\beta, r, 0, sk)$	$FE.Enc(FE.Pk, (C(M_\beta), 0^\kappa, \perp, 0^\kappa); F(r, C))$
H_3^β	$(M_\beta, r', 1, sk')$	$FE.Enc(FE.Pk, (C(M_\beta), 0^\kappa, \perp, 0^\kappa); F(r, C))$
H_4^β	$(M_\beta, r', 1, sk')$	$FE.Enc(FE.Pk, (C(M_\beta), 0^\kappa, \perp, 0^\kappa); F(r, C))$

4.2 Succinct Functional Encryption and Applications

We discuss here the main results of [5]. The primary contribution that the authors present is what they call a "succinct" functional encryption scheme for generalized functions where succinctness refers to having the size of the ciphertexts depend only on the depth of the circuit to be evaluated during decryption and independent of the *size* of the circuit. From this result, the authors use this construction as a building block to several applications including reusable garbled circuits and token-based obfuscation. We first describe their functional encryption scheme and provide a sketch of the proof, then briefly touch upon applications and how they are achieved using FE.

4.2.1 FE Construction

First we give an overview and the intuition behind the construction of this FE scheme then specify the algorithms in detail. We start by recounting the similarities shared between FHE and FE. In particular FHE has an algorithm (the evaluation algorithm) that changes a ciphertext into one such that it decrypts to some function of the encrypted message, whilst FE has an algorithm (the decryption algorithm) that decrypts any given ciphertext into a function f of the encrypted message given a token (or secret key) associated with f. Thus, it would be natural to assume that we can start with an FHE scheme to somehow get FE. That is, the sender would encrypt a message x then send it over to the receiver who will evaluate the function f over the encrypted input to get a ciphertext $\widetilde{f(x)}$ that decrypts to $f(x)$. However, in order for the receiver to decrypt $\widetilde{f(x)}$, it would

need the FHE secret key, and supplying it this secret key would allow it to decrypt everything, including the original ciphertext to get x!

Thus, the next step would be to give the receiver the ability to decrypt the evaluated ciphertext without having to actually know the secret key. To achieve this, one could garble the FHE decryption circuit (with the secret key hardcoded) and then provide this garbled circuit (with the associated input labels) to the receiver. However, in order to preserve the circuit privacy property provided by garbling, we should only give the receiver the input labels associated with $\widetilde{f(x)}$, which is not known by the sender beforehand (since the receiver has yet to apply or decide upon what f to apply). Giving all the input labels to the receiver compromises the security of the garbled circuit and risks revealing the FHE secret key.

To rectify the previous issue, the last step would be to use a 2-outcome attribute-based encryption (ABE) scheme and provide the receiver with all of the garbled circuit's input labels in *encrypted* format as part of the ciphertext. After the receiver requests a secret ABE token SK_f associated with function f, it will be able to recover (only) the labels for $\widetilde{f(x)}$ by using ABE decryption on the encrypted input labels, and then use these labels as input to the garbled FHE decryption circuit to get $f(x)$.

Given this high-level overview, we present the authors' construction of the FE scheme. The components of the scheme consists of a leveled FHE scheme ($\mathsf{FHE}.\mathrm{Setup}$, $\mathsf{FHE}.\mathrm{Enc}$, $\mathsf{FHE}.\mathrm{Eval}$, $\mathsf{FHE}.\mathrm{Dec}$), a 2-outcome (leveled) ABE scheme ($\mathsf{ABE}_2.\mathrm{Setup}$, $\mathsf{ABE}_2.\mathrm{Keygen}$, $\mathsf{ABE}_2.\mathrm{Enc}$, $\mathsf{ABE}_2.\mathrm{Dec}$), and a garbling scheme ($\mathsf{Gb}.\mathrm{Enc}, \mathsf{Gb}.\mathrm{Garble}, \mathsf{Gb}.\mathrm{Eval}$).

Construction 4.4. Let κ be the security parameter, and let $\lambda = \mathrm{poly}(\kappa)$ be the length of the ciphertexts of the leveled FHE scheme.

- $\mathsf{FE}.\mathrm{Setup}(1^\kappa)$:
  1. For all $i \in [\lambda]$ get $(\mathsf{fmpk}_i, \mathsf{fmsk}_i) \leftarrow \mathsf{ABE}_2.\mathrm{Setup}(1^\kappa)$
  2. Output $\mathsf{MPK} := (\mathsf{fmpk}_1, ..., \mathsf{fmpk}_\lambda)$ and $\mathsf{MSK} := (\mathsf{fmsk}_1, ..., \mathsf{fmsk}_\lambda)$
- $\mathsf{FE}.\mathrm{Keygen}(\mathsf{MSK}, f)$: Let f be an n-bit input function. Define $\mathsf{FHE}.\mathrm{Eval}_f^i : \{0,1\}^{|\mathsf{hpk}|} \times \{0,1\}^{n\lambda} \to \{0,1\}$ to be the Boolean function that runs the FHE evaluation algorithm over public key hpk, function f, and n FHE ciphertexts $hc_1, ..., hc_n$ (where each $hc_i \in \{0,1\}^\lambda$ is encrypted under hpk), then outputs the i^{th} bit of the result. Then perform the following steps:
  1. For all $i \in [\lambda]$ get $\mathsf{fsk}_i \leftarrow \mathsf{ABE}_2.\mathrm{Keygen}(\mathsf{MSK}, \mathsf{FHE}.\mathrm{Eval}_f^i)$
  2. Output $\mathsf{SK}_f := (\mathsf{fsk}_1, ..., \mathsf{fsk}_\lambda)$, the secret token for function f
- $\mathsf{FE}.\mathrm{Enc}(\mathsf{MPK}, x)$: Let $x = (x_1, ..., x_n)$ be an n-bit message. Perform the following steps to encrypt x:
  1. $(\mathsf{hpk}, \mathsf{hsk}) \leftarrow \mathsf{FHE}.\mathrm{Setup}(1^\kappa)$
  2. For all $i \in [n]$, get $hc_i \leftarrow \mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, x_i)$. Let $hc = (hc_1, ..., hc_n)$.
  3. Garble the FHE decryption circuit to get $\left(G, \{L_i^0, L_i^1\}_{i=1}^\lambda\right) \leftarrow \mathsf{Gb}.\mathrm{Garble}(\mathsf{FHE}.\mathrm{Dec}(\mathsf{hsk}, .))$ where G is the garbled decryption circuit (with hsk hardcoded), and L_i^b is the label for input x_i with value $b \in \{0,1\}$

4. For all $i \in [\lambda]$, encrypt (L_i^0, L_i^1) with the 2-outcome ABE scheme with respect to attribute (hpk, hc). That is, let $c_i \leftarrow \mathsf{ABE}_2.\mathrm{Enc}(\mathsf{fpk}, (\mathsf{hpk}, hc), L_i^0, L_i^1)$ for all $i \in [\lambda]$
5. Output ciphertext $C = (G, c_1, ..., c_\lambda)$

- $\mathsf{FE}.\mathrm{Dec}(\mathsf{SK}_f, C)$: Parse $C = (G, c_1, ..., c_\lambda)$ and $\mathsf{SK}_f = (\mathsf{fsk}_1, ..., \mathsf{fsk}_\lambda)$. Perform the following steps to decrypt C:

  1. For all $i \in [\lambda]$, recover the labels of the garbled circuit. That is, get $L_i^{k_i} = \mathsf{ABE}_2.\mathrm{Dec}(\mathsf{fsk}_i, c_i)$, where $k_i = \mathsf{FHE}.\mathrm{Eval}_f^i(\mathsf{hpk}, hc)$.
  2. Evaluate the garbled circuit on the recovered labels to get $\mathsf{Gb}.\mathrm{Eval}(G, L_1^{k_1}, ..., L_\lambda^{k_\lambda}) = \mathsf{FHE}.\mathrm{Dec}(\mathsf{hsk}, k_1, ..., k_\lambda) = f(x)$

4.2.2 Proof Sketch

Correctness. To show correctness, we should examine the result of the decryption algorithm $\mathsf{FE}.\mathrm{Dec}(\mathsf{SK}_f, C)$ for a given SK_f and ciphertext $C = \mathsf{FE}.\mathrm{Enc}(\mathsf{MPK}, x) = (G, c_1, ..., c_\lambda)$ where $c_i = \mathsf{ABE}_2.\mathrm{Enc}(\mathsf{fpk}, (\mathsf{hpk}, hc), L_i^0, L_i^1)$ for all $i \in [\lambda]$. In the first step of decryption, we invoke the correctness of the 2-outcome ABE scheme and get the labels of $\mathsf{FHE}.\mathrm{Eval}_f^i(\mathsf{hpk}, hc)$ because the key SK_f contains the tokens for evaluating predicates $\mathsf{FHE}.\mathrm{Eval}_f^i$ and, applying it on the ABE ciphertexts with attribute $(\mathsf{hpk}, hc) = (\mathsf{hpk}, \mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, x))$, we decrypt to get:

$$\mathsf{ABE}_2.\mathrm{Dec}(\mathsf{fsk}_i, c_i) = \begin{cases} L_i^0, & \text{if } \mathsf{FHE}.\mathrm{Eval}_f^i(\mathsf{hpk}, \mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, x)) = 0 \\ L_i^1, & \text{if } \mathsf{FHE}.\mathrm{Eval}_f^i(\mathsf{hpk}, \mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, x)) = 1 \end{cases}$$

In the second step of the decryption algorithm, we use the recovered labels of $\mathsf{FHE}.\mathrm{Eval}_f^i(\mathsf{hpk}, hc)$ as input to the garbled circuit and get:

$$\mathsf{Gb}.\mathrm{Eval}(G, L_1^{k_1}, ..., L_\lambda^{k_\lambda}) \overset{(1)}{=} \mathsf{FHE}.\mathrm{Dec}(\mathsf{hsk}, k_1, ..., k_\lambda) \overset{(2)}{=} f(x)$$

where equality (1) follows from the correctness of garbled circuit evaluation and equality (2) follows from the correctness of the FHE scheme.

Security. The authors use a simulation proof technique to prove that their scheme is simulation-based fully (as opposed to selectively) secure in a non-adaptive single-key setting where the adversary only asks for a single token before the challenge ciphertext is provided [1]. To prove that, they show the existence of a PPT simulator Sim (that does not know the the challenge message x to encrypt) that outputs $\widetilde{c}$ such that the real and ideal worlds are computationally indistinguishable. Intuitively, the simulator will run a modified version of $\mathsf{FE}.\mathrm{Enc}$ to hide the fact that it does not know x. Given $(\mathsf{MPK}, \mathsf{SK}_f, f, f(x), 1^\kappa)$, the simulator Sim works as follows:

1. Run $(\mathsf{hpk}, \mathsf{hsk}) \leftarrow \mathsf{FHE}.\mathrm{Setup}(1^\kappa)$
2. Using the FHE scheme, encrypt n zero-bits to get $\widetilde{0} := \{\mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, 0)_1, ..., \mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, 0)_n\}$

[1] Note that there exists an impossibility result for the case of *adaptive* simulation-based secure functional encryption [2]

3. Using the simulator for the garbling scheme $\mathsf{Sim}_{\mathsf{Gb}}$, garble the $\mathsf{FHE}.\mathrm{Dec}(\mathsf{hsk}, .)$ circuit to get $(G_{\mathsf{Sim}}, \{L_{i,\mathsf{Sim}}\}_{i=1}^{\lambda})$, the simulated garbled circuit and associated labels.

4. Encrypt the labels using the 2-outcome ABE scheme (where the same label is used for either outcome) to get $c_{i,\mathsf{Sim}} = \mathsf{ABE}.\mathrm{Enc}(\mathsf{fmpk}_i, (\mathsf{hpk}, \widetilde{0}), L_{i,\mathsf{Sim}}, L_{i,\mathsf{Sim}})$ for all $i \in [\lambda]$.

5. Output simulated ciphertext $C_{\mathsf{Sim}} = (G_{\mathsf{Sim}}, c_{1,\mathsf{Sim}}, ..., c_{\lambda,\mathsf{Sim}})$

We sketch the proof at a high-level by going over the hybrids and stating the security property used to preserve indistinguishability between any two consecutive hybrids. We start with the first hybrid H_0, which is the ideal experiment using the simulator shown above. The next hybrid H_1 is the same as H_0 except that the simulator in Step 4 will use the ABE scheme to encrypt the labels under the attribute $hc = (\mathsf{hpk}, \mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, x_1), ..., \mathsf{FHE}.\mathrm{Enc}(\mathsf{hpk}, x_n))$ instead of $(\mathsf{hpk}, \widetilde{0})$. Given that FHE is semantically secure, H_0 and H_1 are indistinguishable.

The next hybrid H_2 is the same as H_1 except that we replace the G_{Sim} simulated garbled circuit of $\mathsf{FHE}.\mathrm{Dec}(\mathsf{hsk}, .)$ in the ciphertext with its actual garbling (i.e. using algorithm $\mathsf{Gb}.\mathrm{Garble}$) thus getting $(G, \{L_i^0, L_i^1\}_{i=1}^{\lambda})$. Also, instead of using the ABE scheme to encrypt the simulated labels, we now use the ABE scheme to encrypt $L_i^{k_i}$ twice where $k_i = \mathsf{FHE}.\mathrm{Eval}_f^i(\mathsf{hpk}, hc)$. That is, the ABE ciphertexts are now $c_{i,\mathsf{Sim}} = \mathsf{ABE}.\mathrm{Enc}(\mathsf{fmpk}_i, (\mathsf{hpk}, hc), L_i^{k_i}, L_i^{k_i})$. Given that the garbled circuit scheme is input-private and circuit-private, H_1 and H_2 are indistinguishable.

The last hybrid H_3 is simply the real experiment where we use the actual scheme instead of a simulator, and therefore, the ABE ciphertexts now encrypt both labels L_i^0 and L_i^1. Given that 2-outcome ABE scheme is secure, H_2 and H_3 are indistinguishable.

4.2.3 Reusable Garbled Circuits

The first application of functional encryption that the authors discuss is *reusable* garbled circuits. Standard garbling mechanisms achieve security (circuit/input privacy) assuming one-time evaluation only, which results in high overhead whenever we want a securely evaluate a specific circuit over multiple different encoded inputs, since we would need to re-garble the circuit every time. The authors use a fully-secure single-key succinct functional encryption scheme to reach a solution whereby a garbled circuit can be evaluated over many encoded inputs without the need for re-garbling, and at the same time, preserving circuit and input privacy.

Using a functional encryption scheme ($\mathsf{FE}.\mathrm{Setup}, \mathsf{FE}.\mathrm{Enc}, \mathsf{FE}.\mathrm{Keygen}, \mathsf{FE}.\mathrm{Dec}$) and public-key encryption scheme with public key pk and secret key sk, the authors describe a way to get a reusable garbled circuit scheme. To garble a circuit C, we use the input-hiding property of FE to hide the circuit as well, and the way to do that is to run $\mathsf{FE}.\mathrm{Keygen}$ on a circuit U_E (where E is the encryption of C under pk) that takes as input sk and x, decrypts E to get C then runs C on x. Thus, encoding an input x for the circuit is as simple as encrypting (sk, x) using the FE scheme, and to evaluate a circuit (which can be performed multiple times on the same garbled circuit), we compute $\mathsf{FE}.\mathrm{Dec}$ on the generated token for U_E and the encoded input (sk, x).

4.2.4 Token-based Obfuscation

Another application is *token-based obfuscation*, which can be seen as an reformulation of reusable garbled circuits. Specifically, reusable garbled circuits can be modelled as a form of "private-key" obfuscation in which one can evaluate a given garbled (read obfuscated) circuit over arbitrarily

many inputs only if one has the encoding of said inputs, and these encoded inputs can only be generated by the holder of the private-key. This stands in contrast to the notion of traditional notions of obfuscation, where one is given an obfuscated circuit and anyone could publicly evaluate this circuit over any input of its choice. In fact, given the impossibility result regarding virtual black-box obfuscation, the private encoding of inputs is indeed necessary to achieve simulation-based security in this model.

From reusable garbled circuits, it is straightforward to see how one can also get token-based obfuscation. To obfuscate a circuit C, we garble C first to get $\widetilde{C}$, then output as the obfuscation of C a new circuit $O_{\widetilde{C}}$ that has $\widetilde{C}$ hardcoded and accepts a token tk_x for input x, so that it can evaluate $\widetilde{C}(tk_x) = C(x)$. Token generation for inputs to obfuscated circuits is done by calling the encoding algorithm of the garbling mechanism. Intuitively, a token-based obfuscation scheme is said to be secure if there exists a simulator that can generate an obfuscated circuit and related encodings without knowing the adversary's chosen C or x.

5 Functional Controllable HE

In this section, give our definition of the FCHE primitive and show one way to construct it.

Definition 5.1 (Functional Controllable Homomorphic Encryption Scheme)**.** Let κ be the security parameter and $(\mathcal{F}, \mathcal{G})$ be a pair of function classes represented as circuits. A *functional controllable homomorphic encryption* (FCHE) scheme for $(\mathcal{F}, \mathcal{G})$ is a tuple of six algorithms ($\mathsf{FCHE.}$ Setup, $\mathsf{FCHE.}$ Keygen, $\mathsf{FCHE.}$ Eval, $\mathsf{FCHE.}$ EvalKeygen, $\mathsf{FCHE.}$ Enc, $\mathsf{FCHE.}$ Dec) defined as follows:

- $\mathsf{FCHE.}\,\mathrm{Setup}(1^\kappa)$: a PPT algorithm that takes as input the security parameter 1^κ then generates and outputs a key pair $(\mathsf{MPK}, \mathsf{MSK})$, where MPK is the master public key and MSK is the master secret key.
- $\mathsf{FCHE.}\,\mathrm{Keygen}(\mathsf{MSK}, f)$: a PPT algorithm that takes as input the master secret key MSK and a function $f \in \mathcal{F}$, then outputs a secret decryption token SK_f for function f.
- $\mathsf{FCHE.}\,\mathrm{EvalKeygen}(\mathsf{MSK}, g)$: a PPT algorithm that takes as input the master secret key MSK and a function $g \in \mathcal{G}$, then outputs a secret evaluation token EK_g for function g.
- $\mathsf{FCHE.}\,\mathrm{Enc}(\mathsf{MPK}, x)$: a PPT algorithm that takes as input the master public key MPK and a message $x \in \{0,1\}^n$ then outputs a corresponding ciphertext c
- $\mathsf{FCHE.}\,\mathrm{Eval}(\mathsf{EK}_g, c)$: given an evaluation token for some single-input function g, this PPT algorithm evaluates g on encrypted input c where, $c = \mathsf{FHE.}\,\mathrm{Enc}(\mathsf{hpk}, x)$.
- $\mathsf{FCHE.}\,\mathrm{Dec}(\mathsf{SK}_f, c)$: a PPT algorithm that takes as input a decryption token SK_f and a ciphertext c then outputs the corresponding message m'

We will mainly focus on a slightly different form of this primitive which we call *restrictive* FCHE that has the same syntax as normal FCHE but differs in the $\mathsf{FCHE.}$ Keygen algorithm, where it also requires as input the ciphertext that is to be decrypted with function f.

Correctness: For sufficiently large security parameter κ and polynomial $n = n(\kappa)$, for all $(f, g) \in (\mathcal{F}, \mathcal{G})$ and inputs $x \in \{0,1\}^n$, the following should hold:

$$\Pr[\mathsf{FCHE}.\,\mathrm{Dec}(\mathsf{SK}_f, c) = f(g(x))] \geq 1 - \mathrm{negl}(n)$$

where the evaluated ciphertext is given by $c \leftarrow \mathsf{FCHE}.\,\mathrm{Eval}(\mathsf{EK}_g, c)$ for $c \leftarrow \mathsf{FCHE}.\,\mathrm{Enc}(\mathsf{MPK}, x)$, the evaluation key is $\mathsf{EK}_g \leftarrow \mathsf{FCHE}.\,\mathrm{EvalKeygen}(\mathsf{MSK}, g)$, the decryption key is $\mathsf{SK}_f \leftarrow \mathsf{FCHE}.\,\mathrm{Keygen}(\mathsf{MSK}, f)$, and the master keys are generated as $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{FCHE}.\,\mathrm{Setup}(1^\kappa)$. The probability is over the randomness of the algorithms used in the experiment.

Definition 5.2 (Restricted FCHE IND-CPA Security)**.** An restricted FCHE scheme is said to be IND-CPA secure if for all PPT adversaries A:

$$\Pr[\text{IND-CPA}_A^{\mathsf{FCHE}}(1^\kappa) = 1] \leq \frac{1}{2} + \mathrm{negl}(\kappa)$$

where $\text{IND-FCHE}_A^{\mathsf{FCHE}}$ is shown in Figure 3, and for each of the queries that A sends to the $\mathsf{FCHE}.\,\mathrm{Keygen}$ and $\mathsf{FCHE}.\,\mathrm{EvalKeygen}$ oracle in Step 2, 5, and 6, it must hold that $f(g(x_0)) = f(g(x_1))$ for the requested functions f, g where g as the identity function is implicitly included.

Experiment $\text{IND-CPA}_A^{\mathsf{FCHE}}(1^\kappa)$:
1. $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FCHE}.\,\mathrm{Setup}(1^\kappa)$
2. $(x_0, x_1) \leftarrow A^{\mathsf{FCHE}.\,\mathrm{EvalKeygen}(\mathsf{MSK},.)}(\mathsf{MPK})$
3. $b \xleftarrow{\$} \{0,1\}$
4. $c_b \leftarrow \mathsf{FCHE}.\,\mathrm{Enc}(\mathsf{MPK}, x_b)$
5. $(f, c') \leftarrow A^{\mathsf{FCHE}.\,\mathrm{EvalKeygen}(\mathsf{MSK},.)}(\mathsf{MPK}, c_b)$
6. $\mathsf{SK}_{f,c'} \leftarrow \mathsf{FCHE}.\,\mathrm{Keygen}(\mathsf{MSK}, f, c')$
7. $b' \leftarrow A(\mathsf{MPK}, \mathsf{SK}_{f,c'})$
8. Output $(b = b')$

Figure 3: The $\text{IND-CPA}_A^{\mathsf{FCHE}}$ Experiment

5.1 Restricted FCHE from CHE and Token-based Obfuscation

We now proceed to present our construction. The intuition behind the scheme is reminiscent of the functional encryption scheme in [4]. In particular, we use a CHE scheme as our foundation to encrypt new messages (thus providing input-privacy) and control evaluation, and use an obfuscation mechanism during decryption to apply function f after decrypting the evaluated ciphertexts with the CHE secret key. The obfuscation mechanism ensures circuit-privacy and thus preserves the secrecy of the secret key of the CHE scheme.

Construction 5.3. Let κ be the security parameter. Let $\mathsf{CHE} = (\mathsf{CHE}.\,\mathrm{Setup}, \mathsf{CHE}.\,\mathrm{Keygen}, \mathsf{CHE}.\,\mathrm{Enc}, \mathsf{CHE}.\,\mathrm{HEval}, \mathsf{CHE}.\,\mathrm{Dec})$ be a controllable homomorphic encryption scheme, and $\mathsf{tO} = (\mathsf{tO}.\,\mathrm{Obfuscate}, \mathsf{tO}.\,\mathrm{Enc})$ be a token-based obfuscation scheme.

- $\mathsf{FCHE}.\,\mathrm{Setup}(1^\kappa)$:

1. Run $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \mathsf{CHE}.\mathrm{Setup}(1^\kappa)$
2. Let U_{csk} be the circuit that takes a function f, a CHE ciphertext c then outputs $f(\mathsf{CHE}.\mathrm{Dec}(\mathsf{csk}, c))$. Now, run $(\mathsf{osk}, \widetilde{U}_{\mathsf{csk}}) \leftarrow \mathsf{tO}.\mathrm{Obfuscate}(1^\kappa)$
3. Output $\mathsf{MSK} = (\mathsf{osk}, \mathsf{csk})$ and $\mathsf{MPK} = (\widetilde{U}_{\mathsf{csk}}, \mathsf{cpk})$

- $\mathsf{FCHE}.\mathrm{EvalKeygen}(\mathsf{MSK}, g)$:
  1. Run $\mathsf{EK}_g \leftarrow \mathsf{CHE}.\mathrm{Keygen}(\mathsf{csk}, g)$
  2. Output EK_g
- $\mathsf{FCHE}.\mathrm{Enc}(\mathsf{MPK}, x)$:
  1. Run $c \leftarrow \mathsf{CHE}.\mathrm{Enc}(\mathsf{cpk}, x)$
  2. Output c
- $\mathsf{FCHE}.\mathrm{Eval}(\mathsf{EK}_g, c)$:
  1. Run $c' \leftarrow \mathsf{CHE}.\mathrm{HEval}(\mathsf{cpk}, \mathsf{EK}_g, c)$
  2. Output c'
- $\mathsf{FCHE}.\mathrm{Keygen}(\mathsf{MSK}, f, c)$: For $f \in \mathcal{F}$, do the following:
  1. Run $\mathsf{SK}_{f,c} = \mathsf{tO}.\mathrm{Enc}((f, c))$
  2. Output the encoded inputs $\mathsf{SK}_{f,c}$
- $\mathsf{FCHE}.\mathrm{Dec}(\mathsf{SK}_{f,c}, c)$:
  1. Run $x' \leftarrow \widetilde{U}_{\mathsf{csk}}(\mathsf{SK}_{f,c})$
  2. Output x'

Given the above scheme, we show that it satisfies both the correctness and security properties of the restricted version of FCHE.

Correctness. To show correctness, we should examine the result of the decryption algorithm $\mathsf{FE}.\mathrm{Dec}(\mathsf{SK}_{f,c}, c)$ for a given $\mathsf{SK}_{f,c}$ and ciphertext $c \leftarrow \mathsf{FCHE}.\mathrm{Eval}(\mathsf{EK}_g, c')$ where $c' = \mathsf{CHE}.\mathrm{Enc}(\mathsf{cpk}, x)$ and EK_g is the evaluation key for function g. First, by the correctness of the token-based obfuscation scheme, we get that $\mathsf{FCHE}.\mathrm{Dec}(\mathsf{SK}_{f,c}, c) = \widetilde{U}(\mathsf{SK}_{f,c}) = U(f, c)$. Since U first decrypts c using the CHE decryption algorithm and the key csk, by the correctness of CHE scheme, we will get $g(x)$. After applying the function f on this result, we get the desired correct output $f(g(x))$.

Security. We prove indistinguishability security (IND-CPA) of the restrictive FCHE in the single-key non-adaptive setting where the adversary only asks for a single decryption token after the challenge ciphertext is provided. In our proof, we will use the simulator for the underlying token-obfuscation scheme $\mathsf{Sim}_{\mathsf{tO}}$ and the fact that the CHE scheme is IND-CPA secure.

Starting with the first hybrid H_0 where x_0 is encrypted, we move to the second hybrid H_1 which is exactly the same as H_0 except that the actual tokens SK_f, which are obfuscated programs, will be replaced with simulated obfuscations (using $\mathsf{Sim}_{\mathsf{tO}}$). Thus, if there exists an adversary A that could

distinguish between H_0 and H_1, we can construct an adversary B the breaks the simulation-based security of tO as follows: B starts by creating the keys for the CHE scheme and building the circuit U_{csk}. It then submits U_{csk} to the tO challenger, which will obfuscate the circuit (real or simulated) and return $\widetilde{U}_{\mathsf{csk}}$. B will then run A, providing it with the master public key $\mathsf{MPK} = (\mathsf{cpk}, \widetilde{U}_{\mathsf{csk}})$. After A provides B with the chosen messages (and any requests for evaluation tokens), A will choose to encrypt one of the messages at random. Next, the request for decryption token will be forwarded to the challenger to get $\mathsf{SK}_{f,c}$. Finally, B outputs whatever A outputs.

The next hybrid H_2 is exactly the same as H_1 except that now x_1 will be encrypted. If there exists an adversary A that could distinguish between H_2 and H_1 then we can construct an adversary B the breaks the IND-CPA security of the CHE as follows: B starts by forwarding the simulated circuit U_{csk} and cpk to A. Any requests by A for evaluation tokens will be forwarded to the challenger. When A submits its chosen messages, they will be forwarded to the challenger, which will return the challenge ciphertext. Next, any requests to for decryption tokens will be simulated by B using $\mathsf{Sim}_{\mathsf{tO}}.\mathrm{Enc}((f, c))$. Finally, B outputs whatever A outputs. The last hybrid H_3 is the same as H_2 except now we revert back to the real obfuscated circuit instead of the simulated one. The proof for that proceeds identically to the one between H_0 and H_1.

6 Future Work

We intend to further investigate the minimal assumptions needed to get unrestricted FCHE, and explore the possibility that FCHE itself might provide us with some level of obfuscation. Furthermore, we would like to extend the existing scheme to work with multiple keys that can be composed during evaluation. This raises the question of whether adding further restrictions on the adversary (e.g. $f(g_1(g_2(x_1)))$ must be equal to $f(g_1(g_2(x_2)))$ for all f and g_1, g_2) can be detrimental to the functionality of the scheme, and if so, to what extent. This may also lead us to a notion of security that is based on non-falsifiable assumptions especially if the challenger in the security game needs to verify that the adversary is not cheating by checking that, for all possible compositions of requested keys, the condition imposed on the adversary was not violated.

References

[1] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption: New perspectives and lower bounds," Cryptology ePrint Archive, Report 2012/468, 2012, http://eprint.iacr.org/.

[2] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography*, ser. Lecture Notes in Computer Science, Y. Ishai, Ed. Springer Berlin Heidelberg, 201ONeill20101, vol. 6597, pp. 253–273. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19571-6_16.

[3] Y. Desmedt, V. Iovino, G. Persiano, and I. Visconti, "Controlled homomorphic encryption: Definition and construction," Cryptology ePrint Archive, Report 2014/989, 2014, http://eprint.iacr.org/.

[4] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," Cryptology ePrint Archive, Report 2013/451, 2013, http://eprint.iacr.org/.

[5] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," in *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '13. New York, NY, USA: ACM, 2013, pp. 555–564. [Online]. Available: http://doi.acm.org/10.1145/2488608.2488678.

[6] A. O'Neill, "Definitional issues in functional encryption," Cryptology ePrint Archive, Report 2010/556, 2010, http://eprint.iacr.org/.