

# Controlled Homomorphic Encryption: Definition and Construction

Yvo Desmedt<sup>1</sup>    Vincenzo Iovino<sup>2</sup>    Giuseppe Persiano<sup>3</sup>    Ivan Visconti<sup>3</sup>

<sup>1</sup> Department of Computer Science, University College London, UK, [yvo.desmedt@ucl.ac.uk](mailto:yvo.desmedt@ucl.ac.uk)

<sup>2</sup> University of Luxembourg<sup>a</sup>, [vincenzo.iovino@uni.lu](mailto:vincenzo.iovino@uni.lu)

<sup>3</sup> Dipartimento di Informatica, University of Salerno, Italy, [{giuper,visconti}@dia.unisa.it](mailto:{giuper,visconti}@dia.unisa.it)

## Abstract

Fully Homomorphic Encryption schemes (FHEs) and Functional Encryption schemes (FUNCTEs) have a tremendous impact in Cryptography both for the natural questions that they address and for the wide range of applications in which they have been (sometimes critically) used.

In this work we put forth the notion of a Controllable Homomorphic Encryption scheme (CHES), a new primitive that includes features of both FHEs and FUNCTEs. In a CHES it is possible (similarly to a FHE) to homomorphically evaluate a ciphertext  $Ct = \text{Enc}(m)$  and a circuit  $C$  therefore obtaining  $\text{Enc}(C(m))$  but *only* if (similarly to a FUNCTE) a token for  $C$  has been received from the owner of the secret key.

We discuss difficulties in constructing a CHES and then show a construction based on any FUNCTE.

**Keywords:** Functional Encryption, Non-malleability, Fully Homomorphic Encryption.

---

<sup>a</sup>Part of this work was done while this author was at the University of Warsaw and was supported by the WELCOME/2010-4/2 grant founded within the framework of the EU Innovative Economy Operational Programme

# 1 Introduction

In this paper we put forth the notion of a *controllable homomorphic encryption* scheme (CHES, in short) that blends together the notion of a fully homomorphic encryption scheme [Gen09a] (FHE, in short) and of a functional encryption scheme [BSW11, GVW12a] (FUNCTE, in short). Specifically, like in a FHE, a CHES-ciphertext of plaintext  $m$  can be homomorphically transformed into a ciphertext of plaintext  $C(m)$ , for every efficiently computable function  $C$ ; on the other hand, like in a FUNCTE, the homomorphic transformation can only be efficiently performed by a party that has a special *token* for function  $C$  that is released by the owner of the master secret key. Except for the token for  $C$ , no other secret information is needed to homomorphically transform a ciphertext according to function  $C$ .

**Non-triviality.** The following scheme is a straightforward (albeit inefficient) construction of a CHES derived from any standard public key encryption scheme  $\mathcal{E} = (\text{GenKey}, \text{Enc}, \text{Dec})$  and any secure signature scheme  $\mathcal{S} = (\text{SigKeyGen}, \text{Sign}, \text{Verify})$ . The public key of the CHES consists of a pair  $(\text{pk}, \text{vk})$  of a randomly generated public key  $\text{pk}$  of  $\mathcal{E}$  and of a randomly generated verification key  $\text{vk}$  of  $\mathcal{S}$ . To encrypt message  $m$ , one simply computes an encryption of  $m$  with respect to key  $\text{pk}$ . The token for function  $C$  is simply a signature  $\sigma_C$  of  $C$  and to homomorphically transform ciphertext  $\text{ct}_0$ , one simply appends an encryption  $\text{ct}_1$  of the pair  $(C, \sigma_C)$  to  $\text{ct}_0$ . The decryption function takes a pair of ciphertexts  $(\text{ct}_0, \text{ct}_1)$ , decrypts both and obtains  $(m, C, \sigma_C)$ . If  $\sigma_C$  is a correct signature of  $C$ , then the decryption function outputs  $C(m)$ ; otherwise, it outputs  $\perp$ .

There is a clear drawback in the above construction: the size of the ciphertext depends on the size of the description of the function  $C$ . In this paper, to avoid triviality, we require ciphertext size and decryption time to be upper bounded by a polynomial of the security parameter and be independent of the function  $C$ . This is the same requirement that makes the construction of a FHE non-trivial [Gen09a].

## 1.1 Contribution

The contribution of this work consists of the following three steps. We introduce and define this new primitive, we discuss some interesting applications, and provide a construction based on any FUNCTE. Our main result is the above last step, indeed we will show a general procedure that builds a CHES starting from a general functional encryption scheme.

**Limitations of FUNCTEs w.r.t. CHESs.** At first, one might think that a CHES is just a special case of a functional encryption scheme: the token to transform an encryption of  $m$  into an encryption of  $C(m)$  is simply a token for the function that first computes  $C(m)$  and then re-encrypts the result. Such a direct construction suffers of two major problems.

One first problem posed by this simple construction is that randomness must be used to construct the resulting ciphertext and this would require a notion of functional encryption for probabilistic functionalities proposed in two independent works. Alwen *et al.* [ABF<sup>+</sup>13] put forward a definition of randomized functional encryption but they are able to construct it only for very restricted classes of functionalities. In another work, Goyal *et al.* [GJKS13] propose functional encryption schemes for randomized functionalities for two different notions of security, both suffering from some limitation. The first one is simulation-based but stated in the *selective* model. This is the best one can hope for simulation-based security since, due to the impossibility result of Agrawal *et al.* [AGVW13a] and Boneh *et al.* [BSW11], for non-selective security it is necessary to put a bound on the number of queries the adversary can ask (see also Gorbunov *et al.* [GVW12b], De Caro *et al.* [DIJ<sup>+</sup>13] and De Caro and Iovino [CI13]). The second definition they propose is indistinguishability-based but is affected by the severe problem of forbidding the

adversary to ask queries for computationally indistinguishable distributions, thus not providing any guarantee of security in applications where the server is provided with a token for the re-encryption function. Therefore, for the scope of our applications the solution of Goyal *et al.* is not satisfactory unless one wants to resort to bounded security (i.e., putting a bound on the number of ciphertexts and tokens generated by the system) that represents a strong limitation. Instead, our approach does not suffer from this problem. In fact, we are able to prove the security of our scheme under a notion of security that (1) is *not* selective, (2) allows the adversary to ask an unbounded number of queries and (3) does allow the adversary to request token for the re-encryption function. We also mention that in the context of verifiable secure outsourcing of computation, Barbosa and Farshim [BF12] have studied the concept *Delegatable Homomorphic Encryption* (DFE) which is conceptually very similar to CHES. The security of their construction of a DFE is based on the existence FUNCTE that are CCA1 secure and on the existence of an FHE (in contrast, we only require IND CPA secure FUNCTE).

Another more serious problem in using functional encryption naively is made evident by looking at the following example. Suppose an adversary obtains a token for the increment function  $C(m) := m + 1$ . Clearly, for every two messages  $m_1 \neq m_2$  the output range of the evaluation of the token for  $C$  are disjoint (as one contains encryptions of  $m_1 + 1$  and the other encryptions of  $m_2 + 1$ ). This makes the security requirement of the functional encryption scheme vacuous. Indeed, security for functional encryption schemes is only with respect to adversaries that obtain tokens for which the two challenge plaintexts give the same result and, quite understandably, no guarantees is given for adversaries that have requested and obtained tokens for which the two challenge plaintexts give different results<sup>1</sup>. Therefore, if the token for the innocent looking increment function is released, all security disappears.

**Tricks to construct a CHES.** We obtain a CHES by solving the two major problems of the above direct construction of a CHES from a FUNCTE. Concerning the first problem, we will make use of pseudorandom functions in order to provide to the evaluation process a pseudorandom string to be used for re-encryption. Concerning the second problem, we exploit the fact that even though the two output ranges are disjoint they are still indistinguishable. Interestingly, a similar observation could be used for functional encryption in order to have a relaxed (and therefore easier to achieve) but still fully meaningful definition.

In sums, our construction considers as starting point the problematic construction described above and will leverage on various techniques in order to obtain the desired security. Our construction is proved secure against an adversary that receives tokens for circuits of his choice after seeing the challenge ciphertext. We leave open the problem of constructing a CHES where the adversary can ask encryption and token queries in any order.

**Targeted malleability.** In a recent paper, Boneh et al. [BSW12] put forward the notion of targeted malleability that generalizes the notion of non-malleability [DDN91] by ensuring that the malleability of an encryption scheme is limited to a set of *legal* functions  $\mathcal{F}$ , specified in the public key. We note that, unlike in CHES, in targeted malleability the set  $\mathcal{F}$  of legal functions is specified during the key-generation phase and then any party can efficiently homomorphically transform any ciphertext according to any function in  $\mathcal{F}$  without receiving any secret information from the owner of the secret key. Thus the two primitives are quite different in scope. Boneh et al. [BSW12] show how to transform any FHE scheme into one that offers targeted malleability based on the existence of succinct non-interactive arguments that are known to exist under non-falsifiable assumptions [Nao03].

---

<sup>1</sup> Here we only consider game-based notions of security as simulation-based ones suffer of more serious limitations [BSW11, AGVW13b, DCIJ<sup>+</sup>13, CI13].

We also notice that in their construction [BSW12], ciphertexts obtained through homomorphic transformations can themselves be transformed again and this process can be repeated up to a constant number  $t$  of times; the value  $t$  must be specified during the generation of the public key that grows with  $t$  but the length of the ciphertexts is independent of  $t$ .

In this work, we do not concentrate on this property and give a construction of CHES in which the mauling procedure can only be applied to ciphertexts output by the encryption procedure. We mention though (and do not elaborate further) that our construction can be modified so that the homomorphic transformation procedure can be applied any constant number of times, starting from a ciphertext generated by the encryption procedure. We stress that in our case this number does not affect the length of the ciphertexts nor the one of the public key.

**Indistinguishability obfuscation.** Recently Garg et al. [GGH<sup>+</sup>13] constructed indistinguishability obfuscation, a powerful primitive that overcomes the limitation of virtual black-box obfuscation, and allows to build functional encryption indistinguishability-secure against adversaries asking an unbounded number of token queries.

For sake of simplicity and generality, in this work we focus on constructing a CHES by relying on functional encryption only. An interesting question that we leave open is whether one can construct improved CHESs by weaker assumptions.

**Application scenarios.** The notion of a CHES finds natural applications in the problem of outsourcing computation on private data to an untrusted server.

In the first scenario we consider a user  $U$  that has one message  $m$  and stores it in encrypted form  $Ct$  on an untrusted server  $S$  using a CHES. At some later point,  $U$  wishes to compute value  $C(m)$  and sends a CHES token  $\text{Tok}_C$  for  $C$  (i.e., the token that when applied on a ciphertext for  $m$  returns a ciphertext for  $C(m)$ ). The server  $S$  applies the token  $\text{Tok}_C$  to  $Ct$  and returns the resulting ciphertext to  $U$ . If the server  $S$  is honest-but-curious, the above scheme guarantees that  $U$  gets the desired result without revealing anything about  $m$  (not even the value  $C(m)$ ). The same would work with a FHE. However, a malicious server  $S$  could just pick an arbitrary value, encrypt it using the CHES and then return the value to  $U$ . Against such a dishonest adversary, we can use the standard trick of adding a MAC as follows.  $U$  sends  $S$  an encryption  $Ct$  of  $m$  and of a random value  $R$  (i.e.,  $Ct$  is an encryption of the concatenation of  $m$  and  $R$ ). To compute the value  $C(m)$ , for some circuit  $C$ ,  $U$  picks an arbitrary value  $x$  and generates a token for the circuit that returns an encryption of  $C(m)$  and of  $F(R, x)$ , where  $F$  is a pseudo-random family of functions<sup>2</sup>. In other words, the token encrypts  $C(m)$  and a MAC of the fact that the right token was used to compute the result. Notice that this simple tweak would *not* give security against malicious servers in the above case based on a FHE and this shows that in some applications CHESs is conceptually stronger than FHEs.

CHES is an extension of public key encryption and thus, quite obviously, tokens can be applied to ciphertexts that are generated by third parties. This is different from other approaches to secure outsourcing of computation (see, e.g., [GGP10]) in which it is implicitly implied that ciphertexts are generated by the same party that will later ask to perform computation on the.

Indeed, in the second scenario we consider a user  $U$  who publishes its CHES public key and then various users  $U_1, \dots, U_k$  use  $U$ 's public key to send to  $S$  encryption  $c_1, \dots, c_k$  of messages  $m_1, \dots, m_k$ . Each time  $U$  wants to compute a function over those encrypted messages, it sends a token  $\text{Tok}$  for circuit  $C$  to  $S$ . Then  $S$  applies  $t$  to each of the above  $k$  ciphertexts therefore obtaining new ciphertexts  $c'_1, \dots, c'_k$  that are then sent to  $U$ .  $U$  decrypts those  $k$  ciphertexts therefore obtaining  $C(m_1), \dots, C(m_k)$ . Notice that  $U$  can perform this final decryption steps

---

<sup>2</sup>The use of a PRF is needed to allow the use of more than one token for the same ciphertext; otherwise, a simple encryption of  $C(m)$  concatenated to  $R$  would be sufficient.

(that can be iterated many times depending on the number of the users) by only running the decryption procedure of a CHES, and this can potentially be efficient (i.e., faster than evaluating  $C$ ). Similarly to the first scenario, a simple tweak can make this application secure against a malicious server  $S$ . However in this case users will have to communicate (non-interactively) to  $U$  the values  $R$  they used when computing ciphertexts.

## 2 Definitions

**Notation.** We will denote by  $\lambda$  the security parameter. A *negligible* function  $\nu(\lambda)$  is a function that is smaller than the inverse of any polynomial in  $\lambda$ , for sufficiently large  $\lambda$ . If  $D$  is a probability distribution, the writing “ $x \leftarrow D$ ” means that  $x$  is chosen according to  $D$ . If  $D$  is a finite set, the writing “ $x \leftarrow D$ ” means that  $x$  is chosen according to uniform probability on  $D$ . If  $q > 0$  is an integer then  $[q]$  denotes the set  $\{1, \dots, q\}$ . If  $A$  is an algorithm then  $A(x_1, x_2, \dots)$  denotes the probability distribution of the output of  $A$  when  $A$  is run on input  $(x_1, x_2, \dots)$  and randomly chosen coin tosses. Instead  $A(x_1, x_2, \dots; R)$  denotes the output of  $A$  when run on input  $(x_1, x_2, \dots)$  and (sufficiently long) coin tosses  $R$ .

All algorithms, unless explicitly noted, are probabilistic polynomial time and all adversaries are modeled by non-uniform polynomial time algorithms. If  $B$  is an algorithm and  $A$  is an algorithm with access to an oracle then  $A^B$  denotes the execution of  $A$  with oracle access to  $B$ . If  $a$  and  $b$  are arbitrary strings, then  $a||b$  denotes the string that represents the delimited concatenation of  $a$  and  $b$ . We assume a standard binary encoding for circuits, so for ease of exposition we define functions with binary inputs and invoke them with circuits as inputs instead of their binary encodings.

For the sake of simplicity, in this work we will only consider circuits that on a  $n$ -bit input compute an  $n$ -bit output. However, this can be generalized to any  $n_I$ -bit input and  $n_O$ -bit output circuit where  $n_I = n_I(\lambda)$ ,  $n_O = n_O(\lambda)$  and  $n_I, n_O$  are polynomials.

### 2.1 Functional Encryption for Circuits

In this paper we use a special FUNCTE, which we call FE4C, that allows to compute any polynomial size circuit (see [GVW12a, GGH<sup>+</sup>13]). We start by giving a definition of FE4C and then we present the security notion that we use in our construction.

**Definition 2.1** [FE4C: Functional Encryption Scheme for Circuits] A *Functional Encryption scheme for Circuits* is a tuple  $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Eval})$  of 4 efficient algorithms with the following syntax:

1.  $\text{FE.Setup}(1^\lambda, 1^n)$  outputs *public* and *master secret* keys  $(\text{Pk}, \text{Msk})$  for *security parameter*  $\lambda$  and *length parameter*  $n$ .
2.  $\text{FE.KeyGen}(\text{Msk}, C)$ , on input a master secret key  $\text{Msk}$  for length parameter  $n$  and an  $n$ -bit input and  $n$ -bit output circuit  $C$ , outputs *token*  $\text{Tok}_C$ .
3.  $\text{FE.Enc}(\text{Pk}, M)$ , on input public key  $\text{Pk}$  for length parameter  $n$  and *plaintext*  $M \in \{0, 1\}^n$ , outputs *ciphertext*  $\text{Ct}$ .
4.  $\text{FE.Eval}(\text{Pk}, \text{Ct}, \text{Tok})$  outputs  $B \in \{0, 1\}^n \cup \{\perp\}$ .

For the *correctness* condition we require that for all  $n$ -bit input and  $n$ -bit output circuits  $C$ , all  $M \in \{0, 1\}^n$ , and for  $(\text{Pk}, \text{Msk}) \leftarrow \text{Setup}(1^\lambda, 1^n)$ ,  $\text{Tok} \leftarrow \text{KeyGen}(\text{Msk}, C)$  and  $\text{Ct} \leftarrow \text{Enc}(\text{Pk}, M)$ , the probability that  $\text{Eval}(\text{Pk}, \text{Ct}, \text{Tok}) \neq C(M)$  is negligible in  $\lambda$ .

We formalize security for a FE4C FE by means of the following game FE-INDCPA-GAME between a challenger  $\mathcal{FEC}$  and an adversary  $\mathcal{A}$  that can issue two types of queries to  $\mathcal{FEC}$ , *encryption queries* and *token queries*. The definition is essentially the one in [BSW11].

FE-INDCPA-GAME $_{\mathcal{A}}^{\text{FE}}(\lambda, n)$

**Setup.**  $\mathcal{F}\mathcal{E}.\mathcal{C}$  generates  $(\text{Pk}, \text{Msk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^n)$ , selects random  $b \in \{0, 1\}$  and runs  $\mathcal{A}$  on input  $\text{Pk}$ .

**Token Query.**  $\mathcal{F}\mathcal{E}.\mathcal{C}$  on input an  $n$ -bit input  $n$ -bit output circuit  $C$ , computes and returns  $\text{Tok} \leftarrow \text{FE.KeyGen}(\text{Msk}, C)$ .

**Encryption Query.**  $\mathcal{F}\mathcal{E}.\mathcal{C}$ , on input a pair  $(m_0, m_1)$  of plaintexts, answers by computing and returning  $\text{FE.Enc}(\text{Pk}, m_b)$ .

**Output of the Game.** Let  $b'$  the output of  $\mathcal{A}$ . Then the game outputs 1 if and only if

1.  $b = b'$ ;
2. for all encryption queries  $(m_0, m_1)$ , it holds that  $m_0$  and  $m_1$  are of the same length;
3. for all token queries  $C$  and for all encryption queries  $(m_0, m_1)$ , it holds that  $C(m_0) = C(m_1)$ .

**Definition 2.2** We say that a FE4C FE is IND-CPA *secure* if for all PPT adversaries  $\mathcal{A}$  there exists a negligible function  $\mu$  such that

$$\text{Prob}[\text{FE-INDCPA-GAME}_{\mathcal{A}}^{\text{FE}}(\lambda) = 1] \leq 1/2 + \mu(\lambda).$$

**Tag-Based Non-Rerandomizable Functional Encryption.** In our main construction we use a special type of FE4C in which ciphertexts cannot be re-randomized. More precisely, we consider *tag-based* FE4C in which the encryption algorithm for  $n$ -bit plaintext  $m$  and security parameter  $\lambda$  takes two extra arguments: a *tag*  $\tau$  from the set  $T_\lambda$  of  $\lambda$ -bit tags and an *auxiliary message* of length  $n_1$ . It is easy to see how any FE4C can be modified to accommodate tags and auxiliary messages at the expenses of increasing the length parameter  $n$ . In a *non-rerandomizable tag-based* FE4C, given a ciphertext for an adversarially chosen plaintext  $m$  and auxiliary message  $\text{aux}$  and a random tag  $\tau$ , no adversary can produce another ciphertext for the same plaintext  $m$  and the same tag. Specifically, we consider the following security game between a challenger  $\mathcal{F}\mathcal{E}.\mathcal{C}$  and adversary  $\mathcal{A}$  that can issue one single encryption query.

RERANDOM $_{\mathcal{A}}^{\text{TFE}}(\lambda, n, n_1)$

**Setup.**  $\mathcal{F}\mathcal{E}.\mathcal{C}$  generates  $(\text{Pk}, \text{Msk}) \leftarrow \text{TFE.Setup}(1^\lambda, 1^n, 1^{n_1})$  and runs  $\mathcal{A}$  on input  $\text{Pk}$ .

**Token Query.**  $\mathcal{F}\mathcal{E}.\mathcal{C}$  on input an  $(n + n_1 + \lambda)$ -bit input  $(n + n_1 + \lambda)$ -bit output circuit  $C$ , computes and returns  $\text{Tok} \leftarrow \text{TFE.KeyGen}(\text{Msk}, C)$ .

**Encryption Query.**  $\mathcal{F}\mathcal{E}.\mathcal{C}$ , on input plaintext  $m$  and auxiliary message  $\text{aux}$ , picks a random tag  $\tau$  from the set  $T_\lambda$  of tags of length  $\lambda$  and returns a ciphertext  $\text{Ct} = \text{TFE.Enc}(\text{Pk}, m, \text{aux}, \tau)$  with tag  $\tau$ .

**Output of the Game.** Let  $\text{Ct}^*$  the output of  $\mathcal{A}$ . Then the game outputs 1 if and only if

1.  $\text{Ct}^* \neq \text{Ct}$ ;
2.  $\text{TFE.Dec}(\text{Msk}, \text{Ct}^*) = (m', \text{aux}', \tau')$  with  $m' = m$  and  $\tau = \tau'$ ;

We say that a tag-based FE4C TFE is non-rerandomizable if for all PPT adversaries  $\mathcal{A}$  and for  $n$  and  $n_1$  polynomially bounded in  $\lambda$  there exists a negligible function  $\mu$  such that

$$\text{Prob}[\text{RERANDOM}_{\mathcal{A}}^{\text{TFE}}(\lambda, n, n_1) = 1] \leq \mu(\lambda).$$

It is easy to see that any tag-based FE4C can be transformed into a non-rerandomizable one by using a secure signature scheme. More precisely, we define the encryption algorithm  $\text{TFE.Enc}$  that encrypts plaintext  $m$  and  $\text{aux}$ , to pick a random pair of  $(\text{vk}, \text{sgk})$  of verification and signing key for a secure signature scheme  $(\text{SigKeyGen}, \text{Sign}, \text{Verify})$  and  $m, \text{aux}$  are encrypted using  $\text{vk}$  as tag obtaining  $\text{Ct}$ . Finally, a signature  $\sigma_{\text{Ct}}$  of  $\text{Ct}$  is computed using the signing key  $\text{sgk}$  and the resulting ciphertext consists of the pair  $(\text{Ct}, \sigma_{\text{Ct}})$ . Tokens for function  $C$  on ciphertext  $(\text{Ct}, \sigma_{\text{Ct}})$  first verify  $\sigma_{\text{Ct}}$  and, if successful, proceed to compute  $C(m, \text{aux})$ . We observe that  $\mathcal{A}$  either changes the verification key (and thus changes the tag) or keeps the same verification key but then it has to sign a new ciphertext or compute a new signature (which would violate the security of the signature scheme).

### 3 Controllable Homomorphic Encryption

In this section we define the notion of a *Controllable Homomorphic Encryption Scheme* (CHES).

Roughly speaking, in a CHES it is possible to homomorphically create a string that will be decrypted as  $C(M)$  on input a ciphertext for  $M$  only if one holds a special token for the circuit  $C$ . Similarly to the compactness requirement of FHE, we require that the length of the string homomorphically computed be independent of the circuit.

**Definition 3.1** A *Controllable Homomorphic Encryption Scheme* (CHES, *in short*) is a tuple  $\text{CHE} = (\text{CHE.Setup}, \text{CHE.KeyGen}, \text{CHE.Enc}, \text{CHE.HEval}, \text{CHE.Dec})$  of efficient algorithms with the following syntax and that enjoys the following property of correctness.

1.  $\text{CHE.Setup}(1^\lambda, 1^n)$  on input the *security parameter*  $\lambda$  and *length parameter*  $n$ , outputs *public* and *master secret* keys  $(\text{Pk}, \text{Msk})$ .
2.  $\text{CHE.KeyGen}(\text{Msk}, C)$  on input master secret key  $\text{Msk}$  for length parameter  $n$  and the description of an  $n$ -bit input and  $n$ -bit output circuit  $C$ , outputs *token*  $\text{Tok}_C$  for circuit  $C$ .
3.  $\text{CHE.Enc}(\text{Pk}, M)$  on input public key  $\text{Pk}$  with length parameter  $n$  and *plaintext*  $M \in \{0, 1\}^n$ , outputs a ciphertext  $\text{Ct}$ .
4.  $\text{CHE.HEval}(\text{Pk}, \text{Ct}, \text{Tok})$  on input public key  $\text{Pk}$  for length parameter  $n$ , a ciphertext  $\text{Ct}$  for plaintext  $M \in \{0, 1\}^n$  and a token  $\text{Tok}$  for circuit  $C$ , outputs a string  $\text{Ct}'$  of size independent of  $C$ .
5.  $\text{CHE.Dec}(\text{Msk}, \text{Ct}'')$  on input the master secret key  $\text{Msk}$  and a string  $\text{Ct}''$  outputs a string  $M \in \{0, 1\}^n \cup \{\perp\}$ .

For the correctness we require that  $(\text{CHE.Setup}, \text{CHE.Enc}, \text{CHE.Dec})$  be an encryption scheme, and that there exists a negligible function  $\mu$  such that for all  $n = \text{poly}(\lambda)$ , for all  $n$ -bit input and  $n$ -bit output circuits  $C$ , and all plaintexts  $M \in \{0, 1\}^n$  it holds that:

$$\Pr [\text{CHE.Dec}(\text{Msk}, \text{HEval}(\text{Pk}, \text{Ct}, \text{Tok}_C)) \neq C(M)] \leq \mu(\lambda)$$

where  $(\text{Pk}, \text{Msk}) \leftarrow \text{Setup}(1^\lambda, 1^n)$ ,  $\text{Tok}_C \leftarrow \text{KeyGen}(\text{Msk}, C)$  and  $\text{Ct} \leftarrow \text{Enc}(\text{Pk}, M)$ .

**Multivariate CHES.** FHE is *multivariate* in the sense that it allows to efficiently combine independently computed ciphertexts for messages  $m_1, \dots, m_l$  into one ciphertext for  $C(m_1, \dots, m_l)$  for any efficiently computable multivariate function  $f$ . On the other hand, functional encryption has been defined and constructed as a *univariate* primitive, in the sense that tokens can be issued only for univariate functions. For example, it is not possible to have a token that computes which among two ciphertexts encrypts the larger value.

Since our notion is inspired by functional encryption, we decided to define CHES as a univariate primitive. Both for FUNCTE and CHES obtaining constructions for the multivariate case is an important open question.

**Composing tokens.** In the definition of a CHES, the output of CHE.HEval is not required to be a valid ciphertext (that is, an output of CHE.Enc) and correctness only requires it to be a valid input for CHE.Dec. This means that the security definition does not necessarily need to tolerate an adversary that receives a token for a circuit  $C$ , an encryption of  $m$  and then computes an encryption of  $C^i(m)$ , for any  $i > 0$ . More in general, the security definition does not have to assume that an adversary is able to compose tokens.

We would like to point out that it is possible to formally define a CHES so that tokens could be composed. The requirement then for a successful adversary would be to output a ciphertext of  $C(m_b)$  for a circuit  $C$  that is not the composition of the ones for which she received tokens. However efficiently proving such a fact could be difficult as it is a CO-NP statement.

Along these lines, we point out that the concept of targeted malleability as implemented in [BSW12] allows composition of homomorphic transformations for a constant and fixed number of times (this allows to go-around CO-NP) using non-falsifiable knowledge extraction assumptions (these are needed to construct succinct extractable arguments that are needed for compactness). We finally point out that our construction can be modified to allow a constant and fixed number of compositions of homomorphic transformations, even though in the paper we do not elaborate further.

Given the above subtleties, from now on we consider a ciphertext as the output of the encryption function. While the output of the evaluation function is just a string.

### 3.1 Security of a CHES

As usual in encryption schemes, there are two flavors to measure the security of a CHES. The most interesting flavor is the non-malleable one, since it captures the idea of controlling the capability of mauling a ciphertext. We will therefore continue with the definition of an NM-CPA CHES, and the interested reader can find in Appendix A the notion of IND-CPA CHES, along with some expected implications concerning this notion.

**NM-CPA security of a CHES scheme.** We now consider a security definition for CHES that is the conceptually equivalent to the notion of NM-CPA security of plain encryption schemes. We formalize this notion of security for a CHES  $\text{CHE} = (\text{CHE.Setup}, \text{CHE.KeyGen}, \text{CHE.Enc}, \text{CHE.HEval}, \text{CHE.Dec})$  by means of games  $\text{CHES-NMCPA-GAME}_{b, \mathcal{A}}^{\text{CHE}}$ , for  $b = 0, 1$ , between an adversary  $\mathcal{A}$  and a challenger  $\text{CHE.C}$ . The adversary  $\mathcal{A}$  receives a randomly generated public key of CHE and can issue two types of queries to  $\text{CHE.C}$ : *encryption queries* and *token queries*. Below we formalize how queries are answered by  $\text{CHE.C}$  and the output of the games.



CHES-NMCPA-GAME $_{b,\mathcal{A}}^{\text{CHE}}(\lambda, n)$

**Setup.**  $\mathcal{CHE.C}$  computes  $(\text{Pk}, \text{Msk}) \leftarrow \text{CHE.Setup}(1^\lambda, 1^n)$  and runs  $\mathcal{A}$  on input  $\text{Pk}$ .

**Token Query.**  $\mathcal{CHE.C}$  replies to a token query for a circuit  $C$  by returning  $\text{Tok}^C \leftarrow \text{CHE.KeyGen}(\text{Msk}, C)$ .

**$i$ -th Encryption Query.**  $\mathcal{CHE.C}$  replies to an encryption query  $(M_0^i, M_1^i)$  with  $M_0^i \neq M_1^i$  and  $|M_0^i| = |M_1^i|$ , by returning  $\text{Ct} \leftarrow \text{CHE.Enc}(\text{Pk}, M_b^i)$ .

**Output of the Game.** Let  $(j, \text{Ct}^*, C)$  be  $\mathcal{A}$ 's output.

If all the following conditions hold:

1.  $\mathcal{A}$  did not issue a token query for circuit  $C'$  that coincides with  $C$  on  $M_b^j$ .
2.  $C(M_0^j) \neq C(M_1^j)$ .
3.  $\text{CHE.Dec}(\text{Msk}, \text{Ct}) = C(M_b^j)$ .
4.  $\text{Ct}^*$  is not a ciphertext obtained as a reply to an encryption query.

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

The above definition captures the fact that the adversary manages to produce (see conditions 3 and 4) a new ciphertext of  $C(M_b^j)$  (for otherwise, the adversary could issue encryption queries for  $(M_0, M_1)$  and for  $(C(M_0), C(M_1))$  and returns the ciphertext obtained as a reply to this second query; in case of a single ciphertext query, the adversary could set  $C$  equal to the identity function and return the ciphertext obtained as a reply to the encryption query  $(M_0, M_1)$ ). For this to be a meaningful achievement, it must be that the circuit  $C$  gives different output for the two challenge plaintexts (see condition 2) (for otherwise, the ciphertext could have been obtained by simply giving in output  $C(M_0^j) = C(M_1^j)$ ). Moreover, it is also required (see condition 1) that the adversary has not asked for a token for a function  $C'$  for which  $C(M_b^j) = C'(M_b^j)$  (for otherwise, the ciphertext could have been obtained by simply applying the token).

**Definition 3.2** A CHES  $\text{CHE}$  is a NM-CPA *secure* CHES if for every PPT adversary  $\mathcal{A}$  for all polynomially bounded  $n = n(\lambda)$  we have that the following two ensembles are indistinguishable

$$\left\{ \text{CHES-NMCPA-GAME}_{0,\mathcal{A}}^{\text{CHE}}(\lambda, n) \right\} \text{ and } \left\{ \text{CHES-NMCPA-GAME}_{1,\mathcal{A}}^{\text{CHE}}(\lambda, n) \right\}.$$

A CHES  $\text{CHE}$  is *single-message* NM-CPA *secure* if it is NM-CPA secure with respect to all PPT adversaries  $\mathcal{A}$  that ask exactly one encryption query.

In App. B we show that any single-message NM-CPA-secure CHES is also NM-CPA-secure.

## 4 CHES from Functional Encryption

In this section, we describe an NM-CPA CHES  $\text{CHE} = (\text{CHE.Setup}, \text{CHE.KeyGen}, \text{CHE.Enc}, \text{CHE.HEval}, \text{CHE.Dec})$ . We stress that our security proof works only for an adversary that is required to first ask for an encryption query and then can ask for tokens.

In the description of  $\text{CHE}$ , we let  $\text{FE} = (\text{FE.Setup}, \text{FE.Enc}, \text{FE.KeyGen}, \text{FE.Eval})$  be an IND-CPA secure non-rerandomizable<sup>3</sup> tag-based FE4C. For a CHES with  $n$ -bit plaintexts and security parameter  $\lambda$ , we use an FE for plaintexts of length  $n$ , auxiliary message of length  $\lambda + 2$ . In

<sup>3</sup>In the proof, we give details of the impact of the transformation of Section 2.1 in case FE is re-randomizable.

addition we let  $\mathcal{F} = \{F(\cdot, \cdot)\}$  be a pseudorandom family of functions  $F(\cdot, \cdot)$  (the first argument is the seed), and  $\text{PKE} = (\text{Setup}, \text{Enc}, \text{Dec})$  be a public-key encryption scheme.

For sake of simplicity, we assume that secret keys of  $\text{PKE}$  with security parameter  $\lambda$  are exactly  $\lambda$ -bit long and that ciphertexts of  $n$ -bit messages computed with respect to public key with security parameter  $\lambda$  have length  $\ell = \ell(\lambda, n)$ . Also, not to overburden notation, we assume that the tag space  $T_\lambda$  of  $\text{FE}$  coincides with the seed space of  $F$  and that they both coincide with  $\{0, 1\}^\lambda$ . In addition, for an  $n$ -bit input and  $n$ -bit output circuit  $C$ ,  $\ell$ -bit string  $s$ , and public key  $\text{FE.Pk}$  of  $\text{FE}$ , we denote by  $C_{s, \text{FE.Pk}}$  the  $(n + 2\lambda + 2)$ -bit input circuit defined as follows:

$$C_{s, \text{FE.Pk}}(M, r, t, \text{sk}) = \begin{cases} \text{FE.Enc}(\text{FE.Pk}, (C(M), 0^\lambda, \perp, 0^\lambda); F(r, C)), & \text{if } t = 0; \\ \text{Dec}(\text{sk}, s), & \text{if } t = 1; \\ \perp, & \text{if } t = \perp; \end{cases}$$

where  $M \in \{0, 1\}^n$ ,  $r, \text{sk} \in \{0, 1\}^\lambda$  and  $t \in \{0, 1, \perp\}$ . In what follows, we will drop  $\text{FE.Pk}$  from  $C_{s, \text{FE.Pk}}$  whenever it is clear from the context and simply write  $C_s$ .

Circuit  $C_s$  takes three types of plaintexts: *regular* plaintexts, corresponding to  $t = 0$ , which are the outputs of the encryption algorithm; *mauled* plaintexts, corresponding to  $t = \perp$ , which are outputs of the application of a token; *trapdoor* plaintexts, corresponding to  $t = 1$ , which are used only in the proof. For ciphertexts carrying regular plaintexts, the circuit  $C_s$  outputs a ciphertext for  $C(m)$  and this captures the correct application (through the  $\text{CHE.HEval}$  algorithm) of a token for circuit  $C$  to an encrypted message. Notice that in this case the resulting ciphertext carries a mauled plaintext. For ciphertexts carrying mauled plaintexts, the circuit  $C_s$  outputs  $\perp$  and this captures the (incorrect) application of a token to an already mauled message. For ciphertexts carrying a trapdoor plaintexts,  $C_s$  outputs a decryption of  $s$  with respect to the secret key that is part of the trapdoor plaintext. In the reduction of an adversary for  $\text{CHE}$  to an adversary for  $\text{FE}$ , trapdoor plaintexts are very useful because they force  $C_s$  to return a value that is independent of the actual input  $M$ , and thus can be used to contradict the security of  $\text{FE}$  (cfr., discussion in the Introduction). To do so the value  $s$  used in the generation of the tokens must be carefully chosen so to be indistinguishable from the ones output by algorithm  $\text{CHE.KeyGen}$ .

Algorithm  $\text{CHE.Setup}(1^\lambda, 1^n)$ .

1. Run algorithm  $\text{FE.Setup}$  on input  $(1^\lambda, 1^{n+2\lambda+2})$  and obtain  $(\text{FE.Pk}, \text{FE.Msk})$ ;
2. run algorithm  $\text{Setup}$  on input  $1^\lambda$  and obtain  $(\text{pk}', \text{sk}')$ ;
3. set  $\text{Pk} = \text{FE.Pk}$  and  $\text{Msk} = (\text{FE.Pk}, \text{FE.Msk}, \text{pk}')$ ;
4. return  $(\text{Pk}, \text{Msk})$ .

Algorithm  $\text{CHE.KeyGen}(\text{Msk}, C)$ .

1. set  $s = \text{Enc}(\text{pk}', \text{FE.Enc}(\text{FE.Pk}, (0^n, 0^\lambda, \perp, 0^\lambda)))$ ;  
notice that  $0^n$  is the plaintext,  $(0^\lambda, \perp)$  constitute the auxiliary message and  $0^\lambda$  the tag.
2. set  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Msk}, C_s)$ ;
3. return  $\text{Tok}$ .

Algorithm  $\text{CHE.Enc}(\text{Pk}, M)$ .

1. Randomly select tag  $r \in \{0, 1\}^\lambda$ ;
2. run algorithm  $\text{Setup}$  on input  $1^\lambda$  and obtain  $(\text{pk}, \text{sk})$ ;
3. set  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M, r, 0, \text{sk}))$ ;  
notice that  $M$  is the plaintext,  $(0, \text{sk})$  is the auxiliary message and  $r$  is the tag.
4. return  $\text{Ct}$ .

Algorithm CHE.HEval(Pk, Ct, Tok) outputs FE.Eval(FE.Pk, Ct, Tok);

Algorithm CHE.Dec(Msk, Ct)

1. Set Tok = FE.KeyGen(FE.Msk, ID) where the circuit ID is defined in the following way:  
 $\text{ID}(x_1, x_2, x_3, x_4) = x_1$ .
2. return FE.Eval(FE.Pk, Ct, Tok).

In our construction we are using the  $r$  component of a plaintext both as a tag and as the seed of the PRF that gives the randomness of the ciphertext resulting from the application of a token (see the definition of  $C_s$ ). In case  $T_\lambda$  does not coincide with the seed space of the PRF then we add another value to be used as a seed of a PRF.

The correctness of the scheme follows by the correctness of FE and by the definition of  $C_s$ . In the next section we prove the following theorem.

**Theorem 4.1** Under the assumption of the existence of an IND-CPA secure functional encryption scheme for all circuits (FE4C), there exists a NM-CPA secure CHES secure against adversaries that ask all encryption queries before the token queries.

A construction of a FE4C that is secure (according to Definition 2.2) when the adversary sees any polynomial number of tokens has recently been given in [GGH<sup>+</sup>13].

#### 4.1 Proof on NM-CPA Security

Here we prove the security of our scheme for adversaries that issues all token queries after the encryption query and issues exactly one encryption query (by Theorem B.1 we get security against multi-message adversaries).

Assume that there exists such an adversary  $\mathcal{A}$  that breaks the security of CHE for parameters  $\lambda$  and  $n$ ; that is, there exists a distinguisher  $\mathcal{D}$  such that, denoted by  $p_b(\lambda, n)$  the probability that  $\mathcal{D}$  outputs 1 when its input is sampled according to CHES-NMCPA-GAME $_{b, \mathcal{A}}^{\text{CHE}}(\lambda, n)$ ,  $p_0(\lambda, n) \geq p_1(\lambda, n) + \mu(\lambda)$  for some non-negligible function  $\mu(\cdot)$ .

Based on  $\mathcal{A}$ , we build adversary  $\mathcal{B}$  for FE for security parameter  $\lambda$  and length parameter  $n + 2\lambda + 2$ .  $\mathcal{B}$  interacts with challenger  $\mathcal{FE.C}$  for FE to which  $\mathcal{B}$  can issue encryption and token queries and runs internal copies of  $\mathcal{A}$  and  $\mathcal{D}$ .

Adversary  $\mathcal{B}$  tricks adversary  $\mathcal{A}$  into believing it is interacting with  $\mathcal{CHE.C}$  by presenting a view that differs indistinguishably by the one offered by  $\mathcal{CHE.C}$  since the challenge ciphertext is trapdoor. Then, from  $\mathcal{A}$ 's output  $\mathcal{B}$  manages to decrypt by obtaining from  $\mathcal{FE.C}$  of a token for a function that is equivalent to decryption when applied to  $\mathcal{A}$ 's output but, nonetheless, gives the same value when applied to a ciphertext for one of the two challenge plaintexts output by  $\mathcal{B}$ . This therefore allows  $\mathcal{B}$  to win in the security game of functional encryption.

Next we formally describe how  $\mathcal{B}$ ,  $\mathcal{FE.C}$ ,  $\mathcal{D}$ , and  $\mathcal{A}$  interact.

**Setup.**  $\mathcal{FE.C}$  randomly selects  $b \leftarrow \{0, 1\}$ , computes  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$  and runs  $\mathcal{B}$  on input FE.Pk.

$\mathcal{B}$  computes  $(\text{pk}', \text{sk}') \leftarrow \text{Setup}(1^\lambda)$ , sets CHE.Pk = FE.Pk and runs  $\mathcal{A}$  on input CHE.Pk.

**Encryption query.** When  $\mathcal{A}$  issues an encryption query for the pair of messages  $(M_0, M_1)$ ,  $\mathcal{B}$  proceeds as follows.  $\mathcal{B}$  selects random  $r_0, r_1 \leftarrow \{0, 1\}^\lambda$ , sets  $m_0 = (M_0, r_0, 1, \text{sk}')$  and  $m_1 = (M_1, r_1, 1, \text{sk}')$ , and issues encryption query  $(m_0, m_1)$  to challenger  $\mathcal{FE.C}$ .

We remind the reader that, for  $b = 0, 1$ ,  $M_b$  is the plaintext,  $r_b$  is the tag and  $(1, \text{sk}')$  is the auxiliary message.

Challenger  $\mathcal{FE.C}$  returns  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, m_b)$  and  $\mathcal{B}$  returns ciphertext Ct to  $\mathcal{A}$ .

**Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ ,  $\mathcal{B}$  proceeds as follows.

$\mathcal{B}$  sets  $m_0^C = (C(M_0), 0^\lambda, \perp, 0^\lambda)$  and  $m_1^C = (C(M_1), 0^\lambda, \perp, 0^\lambda)$ , issues encryption query  $(m_0^C, m_1^C)$  to challenger  $\mathcal{FEC}$  and receives  $\text{Ct}^C$  as a reply.

We remind the reader that, for  $b = 0, 1$ ,  $C(M_b)$  is the plaintext,  $0^\lambda$  is the tag and  $(\perp, 0^\lambda)$  is the auxiliary message.

$\mathcal{B}$  then sets  $s = \text{Enc}(\text{pk}', \text{Ct}^C)$ , issues a token query for  $C_s$  and receives token  $\text{Tok}^C$  as a reply.  $\mathcal{B}$  returns  $\text{Tok}^C$  to  $\mathcal{A}$ .

**Output.**  $\mathcal{A}$  outputs circuit  $G$  and  $\text{Ct}^*$  (claimed to be an encryption of  $G(M_b)$ ).

Define circuit  $\tilde{\text{ID}}(\cdot, \cdot, \cdot, \cdot)$  as

$$\tilde{\text{ID}}(M, r, t, \text{sk}) = \begin{cases} \perp, & \text{if } (M, r) = (M_0, r_0) \text{ or } (M, r) = (M_1, r_1); \\ \perp, & \text{if } t = \perp; \\ M, & \text{otherwise;} \end{cases}$$

$\mathcal{B}$  issues token query for  $\tilde{\text{ID}}$  and obtains token  $\text{Tok}^{\tilde{\text{ID}}}$  as a reply from  $\mathcal{FEC}$ .  $\mathcal{B}$  then computes  $\text{Out} = \text{FE.Eval}(\text{FE.Pk}, \text{Ct}^*, \text{Tok}^{\tilde{\text{ID}}})$ . If  $\mathcal{D}(\text{Out}) = 1$  then  $\mathcal{B}$  outputs 0 as its guess for  $b$ ;  $\mathcal{B}$  outputs 1 otherwise.

This ends the description of  $\mathcal{B}$ .

**Handling re-randomizable FE4C.** Before proceeding further, we briefly discuss the impact on algorithm  $\mathcal{B}$  of the transformation outlined in Section 2.1 in case the underlying FE4C FE is re-randomizable. We remind the reader that, in order to enforce non-rerandomizability, the encryption algorithm is modified by using a signature verification key as tag and then adding a signature to the ciphertext. Tokens check that the signature that is part of the ciphertext is correct according to the verification key in the plaintext. Thus we modify the encryption algorithm so that one extra slot is used in the plaintext:  $(M, r, t, \text{sk}, \text{vk})$  where now  $(M, r)$  constitute the plaintext,  $(t, \text{sk})$  the auxiliary message, and  $\text{vk}$  the tag. During the setup, algorithm  $\mathcal{B}$  picks a pair  $(\text{vk}', \text{sgk}')$  and the verification key  $\text{vk}'$  is used as tag in the ciphertexts returned as replies to encryption queries and the associated signing key  $\text{sgk}'$  is used to sign the ciphertexts. The function  $\text{ID}$  is then modified to return  $\perp$  if  $(M, r, \text{vk}) = (M_0, r_0, \text{vk}')$  or if  $(M, r, \text{vk}) = (M_1, r_1, \text{vk}')$ . We omit further details.

**Sanity check.** Let us verify what happens if  $\mathcal{A}$  asks for a token  $\text{Tok}^C$  for circuit  $C$  and then sets  $G = C$  and computes ciphertext  $\text{Ct}^*$  by applying  $\text{Tok}^C$  to the ciphertext  $\text{Ct}$  of  $M_b$  obtained as a reply to the encryption query  $(M_0, M_1)$ . Obviously, such an adversary, by the correctness of CHE, always succeeds but cannot be considered a successful adversary according to the definition of NM-CPA security. Therefore  $\mathcal{B}$  should not be able to break the IND-CPA security of the underlying functional encryption scheme FE, otherwise we would get a contradiction.

In fact, by applying  $\text{Tok}^C$  on  $\text{Ct}$ ,  $\mathcal{A}$  gets a ciphertext for  $(C(M_b), 0^\lambda, \perp, 0^\lambda)$  which is useless since  $\tilde{\text{ID}}(C(M_b), 0^\lambda, \perp, 0^\lambda) = \perp$ , independently from  $b$ . One might think of modifying the definition of  $\text{ID}$  by removing the clause concerning the case  $t = \perp$ . But then  $\mathcal{B}$  would cease to be a legitimate adversary for the IND-CPA security of FE. Indeed, observe that, in the construction of  $\text{Tok}^C$ ,  $\mathcal{B}$  issues encryption queries for plaintexts  $(C(M_0), 0^\lambda, \perp, 0^\lambda)$  and  $(C(M_1), 0^\lambda, \perp, 0^\lambda)$ ; this adds the constraint that  $\mathcal{B}$  can only ask tokens for functions for which these two plaintexts give the same output.

Hybrid	plaintext for Ct	$s$ in Tok
$H_0^\beta$	$(M_\beta, r, 0, \text{sk})$	$\text{Enc}(\text{pk}', \text{FE.Enc}(\text{FE.Pk}, (0^n, 0^\lambda, \perp, 0^\lambda); z))$
$H_1^\beta$	$(M_\beta, r, 0, \text{sk})$	$\text{Enc}(\text{pk}', \text{FE.Enc}(\text{FE.Pk}, (0^n, 0^\lambda, \perp, 0^\lambda); F(r', C)))$
$H_2^\beta$	$(M_\beta, r, 0, \text{sk})$	$\text{Enc}(\text{pk}', \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); F(r, C)))$
$H_3^\beta$	$(M_\beta, r', 1, \text{sk}')$	$\text{Enc}(\text{pk}', \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); F(r, C)))$
$H_4^\beta$	$(M_\beta, r', 1, \text{sk}')$	$\text{Enc}(\text{pk}, \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); z))$

Figure 1: The hybrid experiments at a glance. The pairs  $(\text{pk}, \text{sk})$  and  $(\text{pk}', \text{sk}')$  and the values  $r$  and  $r'$  are randomly chosen at the start of the hybrid experiment. The values  $z$  used as randomness for  $\text{FE.Enc}$  are randomly and independently chosen for each token query. Invocations of the  $\text{Enc}$  algorithm take fresh and independent randomness (not shown above).

We continue the proof by showing that  $\mathcal{B}$  is a legitimate adversary for FE. That is, we need to prove that for all the encryption queries  $(m_0, m_1)$  issued by  $\mathcal{B}$  we have that  $m_0$  and  $m_1$  have the same length; and that, if  $\mathcal{B}$  has issued token query for circuit  $C$  then for all encryption queries  $(m_0, m_1)$  we have that  $C(m_0) = C(m_1)$ . The first condition is easily seen to be satisfied. Let us verify that the second condition holds. For each token query for circuit  $C$  issued by  $\mathcal{A}$ ,  $\mathcal{B}$  issues token query for circuit  $C_s$ .  $\mathcal{B}$ , on the other hand, issues encryption queries for  $(M_0, r_0, 1, \text{sk}')$  and  $(M_1, r_1, 1, \text{sk}')$  while answering  $\mathcal{A}$ 's challenge query. In this case we have  $C_s(M_0, r_0, 1, \text{sk}') = C_s(M_1, r_1, 1, \text{sk}') = \text{Decrypt}(s, \text{sk}')$ .  $\mathcal{B}$  also issues encryption queries while preparing the answer to  $\mathcal{A}$ 's token queries. In this case we have that  $C_s(C(M_0), 0^\lambda, \perp, 0^\lambda) = C_s(C(M_1), 0^\lambda, \perp, 0^\lambda) = \perp$ . A similar reasoning holds for the token query for  $\text{ID}$  issued by  $\mathcal{B}$  in the output phase.

In the rest of the proof we will show that  $\mathcal{A}$ 's view in the interaction with  $\mathcal{B}$  is indistinguishable from the view of  $\mathcal{A}$  in  $\text{CHES-NMCPA-GAME}_{b, \mathcal{A}}^{\text{CHE}}(\lambda, n)$  ( $b$  is the random bit selected by  $\text{CHE.C}$  in the Setup phase). We will then prove that, except with negligible probability, the value  $x$  computed by  $\mathcal{B}$  is the same as the output of  $\text{CHES-NMCPA-GAME}_{b, \mathcal{A}}^{\text{CHE}}(\lambda, n)$  and thus the output of the distinguisher  $\mathcal{D}$  can be used to correctly guess  $b$ . Specifically, we show that, for  $\beta = 0, 1$ , the view of  $\mathcal{A}$  in  $\text{CHES-NMCPA-GAME}_{\beta, \mathcal{A}}^{\text{CHE}}$  is indistinguishable from the view of  $\mathcal{A}$  in the interaction with  $\mathcal{B}$  that is in turn interacting with  $\mathcal{FE.C}$  that sets  $b = \beta$ . We do so by considering a sequence of hybrid experiments and then showing that adjacent hybrid experiments are indistinguishable (see Figure 1).

### Hybrid $H_0^\beta$ .

- Setup.** Set  $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$ ,  $(\text{pk}', \text{sk}') \leftarrow \text{Setup}(1^\lambda)$  and  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$ . Randomly pick  $r, r' \leftarrow \{0, 1\}^\lambda$ . Run  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
- Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ , return  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M_\beta, r, 0, \text{sk}))$ .
- Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ , proceed as follows. Pick random  $z \in \{0, 1\}^\lambda$  and set  $s' = \text{FE.Enc}(\text{FE.Pk}, (0^n, 0^\lambda, \perp, 0^\lambda); z)$ . Set  $s = \text{Enc}(\text{pk}', s')$ .

Set  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Sk}, C_s)$  and return  $\text{Tok}$ .

For  $\beta = 0, 1$ , the view of  $\mathcal{A}$  in  $H_0^\beta$  is the same as the view of  $\mathcal{A}$  is  $\text{CHES-NMCPA-GAME}_{\beta, \mathcal{A}}^{\text{CHE}}(\lambda, n)$ .

**Hybrid  $H_1^\beta$ .** Hybrid  $H_1^\beta$  differs from  $H_0^\beta$  only in the way value  $s'$  is computed in the reply to token query for circuit  $C$ . Specifically,  $s'$  is computed by using pseudorandom value  $F(r', C)$  instead of truly random value  $z$ . The formal description of  $H_1^\beta$  follows.

1. **Setup.** Set  $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$ ,  $(\text{pk}', \text{sk}') \leftarrow \text{Setup}(1^\lambda)$  and  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$ . Randomly pick  $r, r' \leftarrow \{0, 1\}^\lambda$ . Run  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
2. **Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ , return  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M_\beta, r, 0, \text{sk}))$ .
3. **Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ , proceed as follows. Set  $s' = \text{FE.Enc}(\text{FE.Pk}, (0^n, 0^\lambda, \perp, 0^\lambda); F(r', C))$ . Set  $s = \text{Enc}(\text{pk}', s')$  and return  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Sk}, C_s)$ .

Next we show that, by the pseudorandomness of  $F$ , the views of  $\mathcal{A}$  in  $H_0^\beta$  and  $H_1^\beta$  are indistinguishable, for  $\beta = 0, 1$ . We do so by constructing an efficient simulator algorithm  $S$  that interacts with adversary  $\mathcal{A}$  and has access to an oracle  $\mathcal{O}$  that can be either random or pseudorandom. Depending on the nature of  $\mathcal{O}$ ,  $S$  produces  $\mathcal{A}$ 's view in  $H_0^\beta$  or in  $H_1^\beta$ . This suffices for proving that  $H_0^\beta \approx_c H_1^\beta$ . Next we describe  $S$ .

1. **Setup.**  $S$  sets  $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$ ,  $(\text{pk}', \text{sk}') \leftarrow \text{Setup}(1^\lambda)$  and  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$ . Then  $S$  randomly picks  $r, r' \leftarrow \{0, 1\}^\lambda$ . Finally,  $S$  runs  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
2. **Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ ,  $S$  returns  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M_\beta, r, 0, \text{sk}))$ .
3. **Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ ,  $S$  proceeds as follows.  $S$  queries  $\mathcal{O}$  on  $C$ , obtains  $z$  and sets  $s' = \text{FE.Enc}(\text{FE.Pk}, (0^n, 0^\lambda, \perp, 0^\lambda); z)$ . Finally,  $S$  sets  $s = \text{Enc}(\text{pk}', s')$  and returns  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Sk}, C_s)$ .

Suppose the oracle  $\mathcal{O}$  is in random mode; that is, all queries are answered with random values. Then it is easy to see that  $\mathcal{A}$ 's view is exactly the same as in  $H_0^\beta$ . On the other hand, suppose the oracle  $\mathcal{O}$  is in pseudorandom mode; that is,  $\mathcal{O}$  picks a random  $r'$  and, upon receiving  $C$ , it replies with  $F(r', C)$ . Then it is easy to see that  $\mathcal{A}$ 's view is exactly the same as in  $H_1^\beta$ .

**Hybrid  $H_2^\beta$ .** Hybrid  $H_2^\beta$  differs from  $H_1^\beta$  again in the randomness used to compute the value  $s'$ . Specifically,  $s'$  is computed by using as randomness the pseudorandom value  $F(r, C)$  instead of  $F(r', C)$ . The formal description of  $H_2^\beta$  follows.

1. **Setup.** Set  $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$ ,  $(\text{pk}', \text{sk}') \leftarrow \text{Setup}(1^\lambda)$  and  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$ . Randomly pick  $r, r' \leftarrow \{0, 1\}^\lambda$ . Run  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
2. **Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ , return  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M_\beta, r, 0, \text{sk}))$ .
3. **Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ , proceed as follows. Set  $s' = \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); F(r, C))$ . Set  $s = \text{Enc}(\text{pk}', s')$  and return  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Sk}, C_s)$ .

Next we show that, by the IND-CPA security of encryption scheme  $\text{Enc}$ , the views of  $\mathcal{A}$  in  $H_1^\beta$  and  $H_2^\beta$  are indistinguishable, for  $\beta = 0, 1$ . We do so by constructing an IND-CPA adversary  $S$  for  $\text{Enc}$  that uses  $\mathcal{A}$  as a subroutine and interacts with a challenger  $\mathcal{C}$  for  $\text{Enc}$ .  $S$  has the property that if  $\mathcal{C}$  answers  $S$ 's encryption queries for  $(s'_0, s'_1)$  by encrypting  $s'_0$  then  $\mathcal{A}$ 's view is exactly the same as in  $H_1^\beta$ ; on the other hand, if  $\mathcal{C}$  answers encryption queries by encrypting  $s'_1$  then  $\mathcal{A}$ 's view is exactly the same as in  $H_2^\beta$ . Thus, if the two views can be distinguished,  $S$  can break the IND-CPA security of  $\text{Enc}$ . Next we describe  $S$ .

1. **Setup.**  $S$  receives  $\text{pk}'$  from  $\mathcal{C}$  and sets  $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$ . Moreover,  $S$  sets  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$  and randomly picks  $r, r' \leftarrow \{0, 1\}^\lambda$ . Finally,  $S$  runs  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
2. **Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ ,  $S$  returns  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M_\beta, r, 0, \text{sk}))$ .
3. **Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ ,  $S$  proceeds as follows.  $S$  sets  $s'_0 = \text{FE.Enc}(\text{FE.Pk}, (0^n, 0^\lambda, \perp, 0^\lambda); F(r', C))$  and  $s'_1 = \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); F(r, C))$  and issues an encryption query to  $\mathcal{C}$  obtaining  $s$ . Finally,  $S$  returns  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Sk}, C_s)$ .

**Hybrid  $H_3^\beta$ .** Hybrid  $H_3^\beta$  differs from  $H_2^\beta$  in the way  $\mathcal{A}$ 's encryption queries are answered. Specifically,  $\text{Ct}$  is a ciphertext of  $(M_\beta, r', 1, \text{sk}')$  instead of  $(M_\beta, r, 0, \text{sk})$ . The formal description of  $H_3^\beta$  follows.

1. **Setup.** Set  $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$ ,  $(\text{pk}', \text{sk}') \leftarrow \text{Setup}(1^\lambda)$  and  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$ . Randomly pick  $r, r' \leftarrow \{0, 1\}^\lambda$ . Run  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
2. **Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ , return  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M_\beta, r', 1, \text{sk}'))$ .
3. **Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ , proceed as follows. Set  $s' = \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); F(r, C))$ . Set  $s = \text{Enc}(\text{pk}', s')$  and return  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Sk}, C_s)$ .

Next we show that, by the IND-CPA security of functional encryption scheme FE, the views of  $\mathcal{A}$  in  $H_2^\beta$  and  $H_3^\beta$  are indistinguishable, for  $\beta = 0, 1$ . We do so by constructing an IND-CPA adversary  $S$  for FE that uses  $\mathcal{A}$  as a subroutine and interacts with a challenger  $\mathcal{FE.C}$  for FE.  $S$  has the property that if  $\mathcal{FE.C}$  answers  $S$ 's encryption query for  $(m'_0, m'_1)$  by encrypting  $m'_0$  then  $\mathcal{A}$ 's view is exactly the same as in  $H_2^\beta$ ; on the other hand, if  $\mathcal{FE.C}$  answers encryption queries by encrypting  $m'_1$  then  $\mathcal{A}$ 's view is exactly the same as in  $H_3^\beta$ . Thus, if the two views can be distinguished,  $S$  can break the IND-CPA security of FE. Next we describe  $S$ .

1. **Setup.**  $S$  receives  $\text{FE.Pk}$  from  $\mathcal{FE.C}$  and sets  $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$ ,  $(\text{pk}', \text{sk}') \leftarrow \text{Setup}(1^\lambda)$ . Moreover,  $S$  randomly picks  $r, r' \leftarrow \{0, 1\}^\lambda$  and runs  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
2. **Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ ,  $S$  sets

$$m_0 = (M_\beta, r, 0, \text{sk}) \text{ and } m_1 = (M_\beta, r', 1, \text{sk}')$$

issues encryption query  $(m_0, m_1)$  to  $\mathcal{FE.C}$ , obtains  $\text{Ct}$  and returns it to  $\mathcal{A}$ .

3. **Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ ,  $S$  proceeds as follows.  $S$  sets

$$s' = \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); F(r, C)) \text{ and } s = \text{Enc}(\text{pk}', s')$$

and issues an encryption query to  $\mathcal{FE.C}$  for a token for circuit  $C_s$ . Finally,  $S$  returns the token  $\text{Tok}$  received from  $\mathcal{FE.C}$  to  $\mathcal{A}$ .

Firstly, we verify that  $S$  is a legal adversary for IND-CPA of FE. Indeed,  $S$  issues one encryption query for  $m_0$  and  $m_1$  and for all circuits  $C_s$  for which  $S$  asks for a token it holds that

$$C_s(m_0) = C_s(m_1) = \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^n, \perp, 0^n); F(r, c)).$$

Clearly, if  $\mathcal{FE.C}$  returns an encryption of  $m_0$  then  $\mathcal{A}$ 's view is exactly as in  $H_2^\beta$ ; if  $\mathcal{FE.C}$  returns an encryption of  $m_1$  then  $\mathcal{A}$ 's view is exactly as in  $H_3^\beta$ .

**Hybrid  $H_4^\beta$ .** Hybrid  $H_4^\beta$  differs from  $H_3^\beta$  for the randomness used to compute  $s'$ . Specifically, in  $H_3^\beta$ ,  $F(r, C)$  is used as randomness for computing  $s'$  whereas in  $H_4^\beta$  true randomness is used. The formal description of  $H_4^\beta$  follows.

1. **Setup.** Set  $(pk, sk) \leftarrow \text{Setup}(1^\lambda)$ ,  $(pk', sk') \leftarrow \text{Setup}(1^\lambda)$  and  $(\text{FE.Pk}, \text{FE.Sk}) \leftarrow \text{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$ . Randomly pick  $r, r' \leftarrow \{0, 1\}^\lambda$ . Run  $\mathcal{A}$  on input  $\text{CHE.Pk} = \text{FE.Pk}$ .
2. **Encryption query.** When  $\mathcal{A}$  issues an encryption query for messages  $(M_0, M_1)$ , return  $\text{Ct} = \text{FE.Enc}(\text{FE.Pk}, (M_\beta, r', 1, sk'))$ .
3. **Token query.** When  $\mathcal{A}$  issues a token query for circuit  $C$ , proceed as follows. Randomly pick  $z \leftarrow \{0, 1\}^\lambda$  and set  $s' = \text{FE.Enc}(\text{FE.Pk}, (C(M_\beta), 0^\lambda, \perp, 0^\lambda); z)$ . Set  $s = \text{Enc}(pk', s')$  and return  $\text{Tok} = \text{FE.KeyGen}(\text{FE.Sk}, C_s)$ .

Notice that, for  $\beta = 0, 1$ , the view of  $\mathcal{A}$  in  $H_4^\beta$  coincides with the view of  $\mathcal{A}$  while interacting with  $\mathcal{B}$ . Moreover, by the pseudorandomness of  $F$ ,  $H_3^\beta$  and  $H_4^\beta$  are indistinguishable.

**$\mathcal{B}$ 's success probability.** Finally, we show that the probability that  $\mathcal{B}$  correctly guesses  $b$  is at least  $1/2 + \mu(\lambda)$  for a non-negligible function  $\mu$ .

All it is left to show is that the string  $\text{Out}$  computed by  $\mathcal{B}$  and then fed as input to  $\mathcal{D}$  is indistinguishable from the output of  $\text{CHES-NMCPA-GAME}_{b, \mathcal{A}}^{\text{CHE}}$ . This is necessary since  $\mathcal{B}$  does not have access to  $\text{Msk}$  and uses a token for function  $\tilde{\text{ID}}$  instead. However, observe that  $\tilde{\text{ID}}$  differs from the decryption function when the plaintext associated with  $\text{Ct}^*$  is of the form  $(M_b, r_b, t, sk)$  or  $(M_{1-b}, r_{1-b}, t, sk)$  for some  $t$  and  $sk$ . In the first case, this means that  $\mathcal{A}$  has managed to re-randomize a ciphertext for  $\text{FE}$  since it has produced a different ciphertext with the same plaintext  $M_b$  and the same tag  $r_b$  as the plaintext received from the encryption query. This, by hypothesis, occurs only with negligible probability. For the second case, observe that  $r_{1-b}$  is a random  $\lambda$ -bit string that is independent from  $\mathcal{A}$ 's view and thus the probability that  $\mathcal{A}$  produces a ciphertext carrying  $r_{1-b}$  is negligible. We can thus conclude that the input provided by  $\mathcal{A}$  to  $\mathcal{D}$  is indistinguishable from its input in  $\text{CHES-NMCPA-GAME}$ . Therefore, by the hypothesis on  $\mathcal{D}$  we can conclude that  $\mathcal{B}$  breaks the IND-CPA security of  $\text{FE}$ .



## References

- [ABF<sup>+</sup>13] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In Martijn Stam, editor, *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, volume 8308 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2013.
- [AGVW13a] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO (2)*, pages 500–518, 2013.
- [AGVW13b] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 500–518. Springer, 2013.
- [BF12] Manuel Barbosa and Pooya Farshim. Delegatable homomorphic encryption with applications to secure outsourcing of computation. In *Topics in Cryptology - CT-RSA 2012*, *Lecture Notes in Computer Science*, pages 296–312. Springer, Berlin, Germany, 2012.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.
- [BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In *ITCS*, pages 350–366, 2012.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (Standard) LWE. In *FOCS*, pages 97–106, 2011.
- [CI13] Angelo De Caro and Vincenzo Iovino. On the power of rewinding simulators in functional encryption. *Cryptology ePrint Archive*, Report 2013/752, 2013. <http://eprint.iacr.org/>.
- [DCIJ<sup>+</sup>13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O’Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 519–535. Springer, 2013.
- [DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, USA, May 6–8, 1991. ACM Press.

- [DIJ<sup>+</sup>13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O’Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 519–535. Springer, 2013.
- [Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. [crypto.stanford.edu/craig](http://crypto.stanford.edu/craig).
- [Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, Maryland, USA, May 31 – June 2, 2009. ACM Press.
- [GGH<sup>+</sup>13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *IACR Cryptology ePrint Archive*, 451, 2013. (To appear in FOCS 2013).
- [GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2010.
- [GJKS13] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. Functional encryption for randomized functionalities. Cryptology ePrint Archive, Report 2013/729, 2013. <http://eprint.iacr.org/>.
- [GVW12a] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.
- [GVW12b] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2012.
- [Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Berlin, Germany.
- [PSV06] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 271–289. Springer, 2006.

## A IND-CPA CHES

**IND-CPA security of a CHES.** We formalize the notion of security equivalent to IND-CPA for a CHES  $\text{CHE} = (\text{CHE.Setup}, \text{CHE.KeyGen}, \text{CHE.Enc}, \text{CHE.HEval}, \text{CHE.Dec})$  by means of game  $\text{CHES-INDCPA-GAME}$  between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{CHE.C}$ . The adversary  $\mathcal{A}$  receives a randomly generated public key of CHE and can issue two types of queries to  $\mathcal{CHE.C}$ : *encryption queries* and *token queries*. Below we formalize how queries are answered by  $\mathcal{CHE.C}$  and what it means for  $\mathcal{A}$  to win the game.

<p><math>\text{CHES-INDCPA-GAME}_{\mathcal{A}}^{\text{CHE}}(\lambda, n)</math></p> <p><b>Setup.</b> <math>\mathcal{CHE.C}</math> computes <math>(\text{Pk}, \text{Msk}) \leftarrow \text{CHE.Setup}(1^\lambda, 1^n)</math>, selects a random <math>b \in \{0, 1\}</math> and runs <math>\mathcal{A}</math> on input <math>\text{Pk}</math>.</p> <p><b>Token Query.</b> <math>\mathcal{CHE.C}</math> replies to a token query for a circuit <math>C</math> by returning <math>\text{Tok}^C \leftarrow \text{CHE.KeyGen}(\text{Msk}, C)</math>.</p> <p><b><math>i</math>-th Encryption Query.</b> <math>\mathcal{CHE.C}</math> replies to encryption query <math>(M_0^i, M_1^i)</math> with <math> M_0^i  =  M_1^i </math>, by returning <math>\text{Ct} \leftarrow \text{CHE.Enc}(\text{Pk}, M_b^i)</math>.</p> <p><b>Output of the Game.</b> Let <math>b'</math> be <math>\mathcal{A}</math>'s output. Return 1 (meaning that <math>\mathcal{A}</math> has won) iff <math>b = b'</math>.</p>
--

**Definition A.1** A CHES CHE is IND-CPA *secure* if for every PPT adversary  $\mathcal{A}$ , there exists a negligible function  $\mu(\cdot)$  such that

$$\Pr \left[ \text{CHES-INDCPA-GAME}_{\mathcal{A}}^{\text{CHE}}(\lambda, n) = 1 \right] \leq 1/2 + \mu(\lambda).$$

A CHES CHE is *single-message* IND-CPA *secure* if it is IND-CPA secure for all PPT adversaries  $\mathcal{A}$  asking exactly one encryption query.

### A.1 Implications

**IND-CPA CHES from LWE.** Noticing that any fully homomorphic encryption scheme [Gen09b] is also an IND-CPA CHES, we have that the results of [BV11, BGV12] prove the following theorem.

**Theorem A.2** Assuming LWE, there exists an IND-CPA-secure CHES.

**NM-CPA CHES  $\Rightarrow$  IND-CPA CHES.** Here we show the natural implication that every NM-CPA CHES is also an IND-CPA CHES.

**Theorem A.3** Any NM-CPA-secure CHES is also IND-CPA-secure.

**Proof:** The proof is by contradiction. Assume there exists a PPT adversary  $\mathcal{A}$  IND-CPA that is able to guess the challenge bit with probability at least  $1/2 + 1/\text{poly}(\lambda)$ , for some polynomial  $\text{poly}(\cdot)$  and consider the following adversary  $\mathcal{A}'$ .  $\mathcal{A}'$  interacts with a challenger for NM-CPA, runs an internal copy of  $\mathcal{A}$  and uses the challenger to answer  $\mathcal{A}$ 's queries. When  $\mathcal{A}$  outputs  $b$ ,  $\mathcal{A}'$  outputs with the triple  $(1, ct^*, C^*)$ , where  $ct^*$  is an encryption of  $C^*(M_b^1)$  and  $C^*$  is a circuit that satisfies the condition of the definition. Specifically,  $C^*(m) = m + i$  where  $i$  is the smallest integer for which  $C^*$  satisfies the condition of the definition (that is  $C^*(M_0^1) \neq C^*(M_1^1)$  and none of the tokens asked by  $\mathcal{A}$  coincides with  $C^*$  on messages  $M_0^1$  and  $M_1^1$ ). Since the number of token queries is polynomially bounded, the circuit  $C^*$  can be efficiently found. ■

**IND-CPA: single message vs many messages.** Here we show that in order to prove the IND-CPA and NM-CPA security of a construction for a CHES, it is sufficient to concentrate on the case of an adversary that asks for one encryption query only. Indeed, we prove that any single-message IND-CPA-secure CHES is also IND-CPA and similarly for NM-CPA security.

**Theorem A.4** Any single-message IND-CPA-secure CHES is also IND-CPA.

**Proof:** Consider a single-message IND-CPA CHES CHE and suppose by contradiction that there exists an adversary  $\mathcal{A}$  against its (many-message) IND-CPA security that succeeds with probability  $1/2 + 1/\text{poly}(\lambda)$  for a polynomial  $\text{poly}(\cdot)$ . We construct an adversary  $\mathcal{A}'$  against the single-message IND-CPA security of CHE as follows.  $\mathcal{A}'$  behaves as a proxy between the challenger of single-message IND-CPA and  $\mathcal{A}$  except for encryption queries and answers. Specifically,  $\mathcal{A}'$  selects a random bit  $b'$  and a random index  $j'$  in  $\{1, \dots, q\}$ , where  $q$  is an upper bound on the number of queries of  $\mathcal{A}$  and will behave as a proxy between  $\mathcal{A}$  and the challenger for all token queries and for  $j'$ -th encryption query; for the remaining encryption queries  $(M_0^i, M_1^i)$  instead,  $\mathcal{A}'$  replies by computing an encryption of  $M_{b'}^i$ . Finally,  $\mathcal{A}'$  outputs the same bit that  $\mathcal{A}$  outputs.

Observe that the success probability of  $\mathcal{A}$  is equal to  $1/2(S_0 + S_1)$ , where we let  $S_b$  denote the success probability of  $\mathcal{A}$  when the challenger chooses bit  $b$ . Therefore we have that  $1/2(S_0 + S_1) \geq 1/2 + 1/\text{poly}(\lambda)$ .

Now consider the probability that  $\mathcal{A}$  outputs the same bit  $b$  chosen by the challenger but in an experiment where for a randomly chosen challenge ciphertext the value  $\bar{b}$  is used instead of  $b$ , and let us denote such a probability by  $T_b$ . Notice that, for  $b = 0, 1$ ,  $T_b \geq S_b - \mu(\lambda)$  for some negligible function  $\mu$ , for otherwise we can trivially break the single-message IND-CPA security of CHE.

Noticing that with probability  $1/2$  it holds that  $b = b'$ , we have that the success probability of  $\mathcal{A}'$  is  $S_0 + S_1$  with probability  $1/2$ , and  $T_0 + T_1$  with probability  $1/2$ .

Summing up, the success probability of  $\mathcal{A}'$  can be computed as follows:  $\frac{S_0 + S_1 + T_0 + T_1}{2} \geq 1/2 + \text{poly}'(\lambda)$  for some polynomial  $\text{poly}'$ . ■

## B Single-Message vs Multi-Message NM-CPA CHES

**Theorem B.1** Any single-message NM-CPA-secure CHES is also NM-CPA-secure.

**Proof:** Let CHE be a single-message NM-CPA-secure CHES. Assume by contradiction that there exists a successful adversary  $\mathcal{A}$  for NM-CPA security and an efficient distinguisher  $\mathcal{D}$  that distinguishes

$$\text{CHES-NMCPA-GAME}_{0,\mathcal{A}}^{\text{CHE}}(\lambda, n) \text{ and } \text{CHES-NMCPA-GAME}_{1,\mathcal{A}}^{\text{CHE}}(\lambda, n).$$

We now reduce  $\mathcal{A}$  to an adversary  $\mathcal{A}'$  for single-message NM-CPA security of CHES. The reduction is similar to the one given in [PSV06]. Let  $q > 1$  be an upper bound on the the number of encryption queries made by  $\mathcal{A}$ . Consider the game  $\text{CHES-NMCPA-GAME}_{J,\mathcal{A}}^{\text{CHE}}(\lambda, n)$  indexed by vector  $J = (b_1, \dots, b_q)$  that specifies that the  $j$ -th encryption query is answered by encrypting  $M_{b_j}^j$ . For  $j = 0, \dots, q$ , we define vector  $J_j = (1, \dots, 1, 0, 0, \dots, 0)$  as the vector whose

first  $j$  components are 1 and the remaining components are 0. We can now run hybrid arguments since  $\text{CHES-NMCPA-GAME}_{0,\mathcal{A}}^{\text{CHE}}(\lambda, n)$  corresponds to  $\text{CHES-NMCPA-GAME}_{J_0,\mathcal{A}}^{\text{CHE}}(\lambda, n)$  and  $\text{CHES-NMCPA-GAME}_{1,\mathcal{A}}^{\text{CHE}}(\lambda, n)$  corresponds to  $\text{CHES-NMCPA-GAME}_{J_q,\mathcal{A}}^{\text{CHE}}(\lambda, n)$ . Since  $\mathcal{D}$  distinguishes

$$\text{CHES-NMCPA-GAME}_{J_0,\mathcal{A}}^{\text{CHE}}(\lambda, n) \text{ and } \text{CHES-NMCPA-GAME}_{J_q,\mathcal{A}}^{\text{CHE}}(\lambda, n),$$

there exists  $j \in \{0, \dots, q-1\}$  such that  $\mathcal{D}$  distinguishes between

$$\text{CHES-NMCPA-GAME}_{J_j,\mathcal{A}}^{\text{CHE}}(\lambda, n) \text{ and } \text{CHES-NMCPA-GAME}_{J_{j+1},\mathcal{A}}^{\text{CHE}}(\lambda, n).$$

We can therefore use  $\mathcal{D}$  along with an adversary  $\mathcal{A}'$  to contradict single-message NM-CPA security of CHE as follows.  $\mathcal{A}'$  behaves as proxy between the challenger and  $\mathcal{A}$  for the token queries. Instead encryption queries are handled as follows.  $\mathcal{A}$  selects a random  $j' \in \{0, \dots, q-1\}$  and forwards to the challenger the  $j'$ -th encryption query  $(M_0^{j'}, M_1^{j'})$  received from  $\mathcal{A}$ , and forwards to  $\mathcal{A}$  the corresponding answer received from the challenger. Instead, for all remaining encryption queries  $(M_0^i, M_1^i)$ ,  $\mathcal{A}'$  answers on its own by sending an encryption of  $M_1^i$  when  $i < j'$  and of  $M_0^i$  when  $i > j'$ .

Assume  $j = j'$ . Notice that when the challenger encrypts  $M_0^j$ , the above game corresponds to

$$\text{CHES-NMCPA-GAME}_{J_j,\mathcal{A}}^{\text{CHE}}(\lambda, n)$$

while when the challenger encrypts  $M_1^j$ , the above game corresponds to

$$\text{CHES-NMCPA-GAME}_{J_{j+1},\mathcal{A}}^{\text{CHE}}(\lambda, n).$$

By conditioning on the event that  $j = j'$  we conclude observing that therefore  $\mathcal{D}$  distinguishes  $\text{CHES-NMCPA-GAME}_{0,\mathcal{A}'}^{\text{CHE}}(\lambda', n)$  from  $\text{CHES-NMCPA-GAME}_{1,\mathcal{A}'}^{\text{CHE}}(\lambda, n)$ .  $\blacksquare$