Controlled Homomorphic Encryption: Definition and Construction

Yvo Desmedt[1] Vincenzo Iovino[2] Giuseppe Persiano[3]
Ivan Visconti[3]

[1] University of Texas at Dallas, USA and University College London, UK,
yvo.desmedt@utdallas.edu
[2] University of Luxembourg, vinciovino@gmail.com
[3] Dipartimento di Informatica, University of Salerno, Italy,
{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 $\mathsf{Ct} = \mathsf{Enc}(m)$ and a circuit C therefore obtaining $\mathsf{Enc}(C(m))$ 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.
As a byproduct our CHES also represents a FunctE supporting the reencryption functionality and in that respect improves existing solutions.

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

1 Introduction

Fully Homomorphic Encryption has received a lot of attention and even was mentioned in the New York Times. We first briefly argue that in many real life applications, Fully Homomorphic Encryption is *not* a very useful primitive. In particular we look at issues involving financial issues and issues that are money related.

For privacy reasons many utility bills and bank statements, sent electronically, are now encrypted. In practice such bills and other documents are based on standard sub-documents that have been carefully checked by the legal department of the utility corporation or the bank. The other parts are based on

the amount due, or the transactions made by the bank's customer. Obviously, very different type of standard letters, checked by the legal department, are sent to customers who are not paying their bills, and in the worst case a disconnect letter is sent.

To maintain privacy as much as possible the original standard letters should be stored on the corporation's company under encrypted form. So, it seems that Fully Homomorphic Encryption is ideally suited to modify the standard letter to include the name and address of the customer and to also add other encrypted information, e.g., obtained from an electronic utility meter. We now explain why this is a bad idea.

Now that Podesta's (the chairman of the 2016 Hillary Clinton presidential campaign) e-mails were hacked many people have realized how vulnerable systems are. Moreover, similar phishing attacks have been used against large corporations. So, in case the computer used to make encrypted utility bills and bank statements is hacked and Fully Homomorphic Encryption is used, *then the hacker can completely change the letter*! Obviously, the legal department does not want to have potentially very offensive letters to be produced. In the case of manual editing of the standard file, digital signature could be used, but if many parties are authorized, then that would require to keep track of all changes made and to keep the digital signatures for these intermediately produced messages. We now briefly motivate our approach.

In our approach, we will avoid the aforementioned use of digital signatures. Instead, using a new primitive, we will restrict what changes can be made. So, no hacker, even having full control of the computer, will be able to drastically change the encrypted document. The worst that can happen is that the wrong (slightly modified) standard letter will be sent. However, in our system, if the status of the customer is maintained under encrypted form, it will even be impossible that the complete wrong standard letter is sent to the customer. Similar protecting mechanisms can be used such that in case of utility bill, the encrypted data provided by the meter has to be used.

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 [15] (FHE, in short) and of a functional encryption scheme [5,17] (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 $\mathcal{E} = (\mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec})$ and any secure signature scheme $\mathcal{S} = (\mathsf{SigKeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The public key of the CHES consists of a pair $(\mathsf{pk}, \mathsf{vk})$ of a randomly generated

public key pk of $\mathcal{E}$ and of a randomly generated verification key vk of $\mathcal{S}$. To encrypt message m, one simply computes an encryption of m with respect to key pk. The token for function C is simply a signature σ_C of C and to homomorphically transform ciphertext ct_0, one simply appends an encryption ct_1 of the pair (C, σ_C) to ct_0. The decryption function takes a pair of ciphertexts $(\mathsf{ct}_0, \mathsf{ct}_1)$, decrypts both and obtains (m, C, σ_C). If σ_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 [15].

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 FunctE*s w.r.t.* CHES*s* 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.

Probabilistic and re-encryption functionalities 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.* [3] 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.* [19] 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.* [1] and Boneh *et al.*[5], 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.* [18], De Caro *et al.*[11] and De Caro and Iovino [10]). 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 [4] 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 IndCPA secure FunctE). As a byproduct our CHES offers a solution to the problem of providing a FunctE supporting the re-encryption functionality that improves the previous works as discussed above.

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[1]. 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. [6] put forward the notion of targeted malleability that generalizes the notion of non-malleability [12] by

[1] Here we only consider game-based notions of security as simulation-based ones suffer of more serious limitations [5,2,11,20,10].

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. [6] 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 [21]. We are aware that such assumptions could ease the construction of a CHES but one of our goals is to avoid them.

We also notice that in their construction [6], 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.

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 $\mathtt{Tok}_C$ 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 $\mathtt{Tok}_C$ 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[2]. In other words,

[2] 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.

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.

As mentioned before, CHES also represents a FUNCTE supporting the re-encryption functionality and in that respect it improves existing solutions as discussed previously.

2 Definitions

Notation

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 [17,13]). Due to space constraints we defer to Appendix A the definition of FE4C and of its tag-based version and its security notion of IND-CPA Security that we use in our construction.

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 1. *A* Controllable Homomorphic Encryption Scheme (CHES, in short) *is a 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 and that enjoys the following property of correctness.*

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

For the correctness we require that $(\mathsf{CHE.Setup}, \mathsf{CHE.Enc}, \mathsf{CHE.Dec})$ *be an encryption scheme, and that there exists a negligible function* μ *such that for all* $n = \mathsf{poly}(\lambda)$*, for all* n*-bit input and* n*-bit output circuits* C*, and all plaintexts* $M \in \{0,1\}^n$ *it holds that:* $\Pr[\,\mathsf{CHE.Dec}(\mathsf{Msk}, \mathsf{HEval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_C)) \neq C(M)\,] \leq \mu(\lambda)$*, where* $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$*,* $\mathtt{Tok}_C \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, C)$ *and* $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, M)$*.*

Composing tokens In the definition of a CHES, the output of $\mathsf{CHE.HEval}$ is not required to be a valid ciphertext (that is, an output of $\mathsf{CHE.Enc}$) and correctness only requires it to be a valid input for $\mathsf{CHE.Dec}$. 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 [6] 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 B 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 $\mathsf{CHE} = (\mathsf{CHE.Setup}, \mathsf{CHE.KeyGen}, \mathsf{CHE.Enc}, \mathsf{CHE.HEval}, \mathsf{CHE.Dec})$ by means of games CHES-NMCPA-$\textsc{Game}^{\mathsf{CHE}}_{b,\mathcal{A}}$, for $b = 0, 1$, between an adversary $\mathcal{A}$ and a challenger $\mathcal{CHE}.\mathcal{C}$. The adversary $\mathcal{A}$ receives a randomly generated public key of

CHE and can issue two types of queries to $\mathcal{CHE}.\mathcal{C}$: *encryption queries* and *token queries*. Below we formalize how queries are answered by $\mathcal{CHE}.\mathcal{C}$ and the output of the games.

CHES-NMCPA-$\textsc{Game}^{\mathsf{CHE}}_{b,\mathcal{A}}(\lambda, n)$

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

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

***i*-th Encryption Query.** $\mathcal{CHE}.\mathcal{C}$ 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 $\mathsf{Ct} \leftarrow \mathsf{CHE.Enc}(\mathsf{Pk}, M_b^i)$.

Output of the Game. Let $(j, \mathsf{Ct}^\star, C)$ 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. $\mathsf{CHE.Dec}(\mathsf{Msk}, \mathsf{Ct}^\star) = C(M_b^j)$.
4. $\mathsf{Ct}^\star$ 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 encryptions 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).[3]

Definition 2. *A* CHES 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* $\{\text{CHES-NMCPA-}\textsc{Game}^{\mathsf{CHE}}_{0,\mathcal{A}}(\lambda, n)\}$ *and* $\{\text{CHES-NMCPA-}\textsc{Game}^{\mathsf{CHE}}_{1,\mathcal{A}}(\lambda, n)\}$. *A* CHES CHE *is* single-message NM-CPA

[3] Another ostensible issue in the definition could be the following. In the the security game, the adversary can make 0 token queries, 1 encryption query and get $\mathsf{Ct} = \mathsf{CHE.Enc}(\mathsf{Pk}, M_b)$, and generates by himself $\mathsf{Ct}' = \mathsf{CHE.Enc}(\mathsf{Pk}, (M_0, r, 0, \mathsf{Sk}))$ for a freshly drawn Sk, and output $(0, \mathsf{Ct}', \mathsf{ID})$. The conditions (1)-(4) of the game hold with probability 1/2. (1) A did not ask any token query. (2) $\mathsf{ID}(M_0) \neq \mathsf{ID}(M_1)$. (3) The decryption works. (4) Ct' has not been obtained by an encryption query (but as a fresh encryption query). The flaw in this reasoning is that the adversary can trivially win with at most probability 1/2.

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

In App. C 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 $\mathsf{CHE} = (\mathsf{CHE.Setup}, \mathsf{CHE.KeyGen}, \mathsf{CHE.Enc}, \mathsf{CHE.HEval}, \mathsf{CHE.Dec})$. 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 CHE, we let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Eval})$ be an IND-CPA secure non-rerandomizable[4] tag-based FE4C. For a CHES with n-bit plaintexts and security parameter λ, we use an FE for plaintexts of length n, auxiliary message of length $\lambda + 2$. In addition we let $\mathcal{F} = \{F(\cdot,\cdot)\}$ be a pseudorandom family of functions $F(\cdot,\cdot)$ (the first argument is the seed), and $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme.

For sake of simplicity, we assume that secret keys of PKE with security parameter λ are exactly λ-bit long and that ciphertexts of n-bit messages computed with respect to public key with security parameter λ have length $\ell = \ell(\lambda, n)$. Also, not to overburden notation, we assume that the tag space T_λ of 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, ℓ-bit string s, and public key $\mathsf{FE.Pk}$ of FE, we denote by $C_{s,\mathsf{FE.Pk}}$ the $(n+2\lambda+2)$-bit input circuit defined as follows:

- if $t = 0$ then $C_{s,\mathsf{FE.Pk}}(M, r, t, \mathsf{sk}) = \mathsf{FE.Enc}(\mathsf{FE.Pk}, (C(M), 0^\lambda, \perp, 0^\lambda); F(r, C))$,
- if $t = 1$ then $C_{s,\mathsf{FE.Pk}}(M, r, t, \mathsf{sk}) = \mathsf{Dec}(\mathsf{sk}, s)$,
- if $t = \perp$ then $C_{s,\mathsf{FE.Pk}}(M, r, t, \mathsf{sk}) = \perp$,

where $M \in \{0,1\}^n$, $r, \mathsf{sk} \in \{0,1\}^\lambda$ and $t \in \{0, 1, \perp\}$.

In what follows, we will drop $\mathsf{FE.Pk}$ from $C_{s,\mathsf{FE.Pk}}$ 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 $\mathsf{CHE.HEval}$ 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

[4] In the proof, we give details of the impact of the transformation of Section 2 in case FE is re-randomizable.

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 CHE to an adversary for 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 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 $\mathsf{CHE.KeyGen}$.

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

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

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

Algorithm $\mathsf{CHE.HEval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$ outputs $\mathsf{FE.Eval}(\mathsf{FE.Pk}, \mathsf{Ct}, \mathtt{Tok})$;

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

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_λ 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 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.*

Constructions of FE4C that are secure (according to Definition 4) when the adversary sees any polynomial number of tokens have recently been given in [13,7,23,14].

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 5 we get security against multi-message adversaries).

Assume that there exists such an adversary $\mathcal{A}$ that breaks the security of CHE for parameters λ 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-$\textsc{Game}^{\mathsf{CHE}}_{b,\mathcal{A}}(\lambda, n)$, $p_0(\lambda, n) \geq p_1(\lambda, n) + \mu(\lambda)$ for some non-negligible function $\mu(\cdot)$.

Based on $\mathcal{A}$, we build adversary $\mathcal{B}$ for FE for security parameter λ and length parameter $n + 2\lambda + 2$. $\mathcal{B}$ interacts with challenger $\mathcal{FE}.\mathcal{C}$ 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 $\mathcal{CHE}.\mathcal{C}$ by presenting a view that differs indistinguishably by the one offered by $\mathcal{CHE}.\mathcal{C}$ since the challenge ciphertext is trapdoor. Then, from $\mathcal{A}$'s output $\mathcal{B}$ manages to decrypt by obtaining from $\mathcal{FE}.\mathcal{C}$ 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 $\mathcal{B}$, $\mathcal{FE}.\mathcal{C}$, $\mathcal{D}$, and $\mathcal{A}$ interact.

Setup. $\mathcal{FE}.\mathcal{C}$ randomly selects $b \leftarrow \{0,1\}$, computes $(\mathsf{FE.Pk}, \mathsf{FE.Sk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{n+2\lambda+2})$ and runs $\mathcal{B}$ on input $\mathsf{FE.Pk}$.
$\mathcal{B}$ computes $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Setup}(1^\lambda)$, sets $\mathsf{CHE.Pk} = \mathsf{FE.Pk}$ and runs $\mathcal{A}$ on input $\mathsf{CHE.Pk}$.

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 $m_0 = (M_0, r_0, 1, \mathsf{sk}')$ and $m_1 = (M_1, r_1, 1, \mathsf{sk}')$, and issues encryption query (m_0, m_1) to challenger $\mathcal{FE}.\mathcal{C}$.
We remind the reader that, for $b = 0, 1$, M_b is the plaintext, r_b is the tag and $(1, \mathsf{sk}')$ is the auxiliary message.
Challenger $\mathcal{FE}.\mathcal{C}$ returns $\mathsf{Ct} = \mathsf{FE.Enc}(\mathsf{FE.Pk}, m_b)$ 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 $\mathcal{FE}.\mathcal{C}$ and receives Ct^C as a reply.
We remind the reader that, for $b = 0, 1$, $C(M_b)$ is the plaintext, 0^λ is the tag and $(\perp, 0^\lambda)$ is the auxiliary message.
$\mathcal{B}$ then sets $s = \mathsf{Enc}(\mathsf{pk}', \mathsf{Ct}^C)$, issues a token query for C_s and receives token $\mathtt{Tok}^C$ as a reply. $\mathcal{B}$ returns $\mathtt{Tok}^C$ to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs circuit G and $\mathsf{Ct}^\star$ (claimed to be an encryption of $G(M_b)$).
Define circuit $\tilde{\mathsf{ID}}(\cdot, \cdot, \cdot, \cdot)$ as follows:
- if $(M, r) = (M_0, r_0)$ or $(M, r) = (M_1, r_1)$ then $\tilde{\mathsf{ID}}(M, r, t, \mathsf{sk}) = \perp$,
- if $t = \perp$ then $\tilde{\mathsf{ID}}(M, r, t, \mathsf{sk}) = \perp$,
- $\tilde{\mathsf{ID}}(M, r, t, \mathsf{sk}) = M$ otherwise.

$\mathcal{B}$ issues token query for $\tilde{\mathsf{ID}}$ and obtains token $\mathtt{Tok}^{\tilde{\mathsf{ID}}}$ as a reply from $\mathcal{FE}.\mathcal{C}$. $\mathcal{B}$ then computes $\mathtt{Out} = \mathsf{FE.Eval}(\mathsf{FE.Pk}, \mathsf{Ct}^\star, \mathtt{Tok}^{\tilde{\mathsf{ID}}})$. If $\mathcal{D}(\mathtt{Out}) = 1$ 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 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: $(M, r, t, sk, \mathsf{vk})$ where now (M, r) constitute the plaintext, (t, sk) the auxiliary message, and vk the tag. During the setup, algorithm $\mathcal{B}$ picks a pair $(\mathsf{vk}', \mathsf{sgk}')$ and the verification key vk' is used as tag in the ciphertexts returned as replies to encryption queries and the associated signing key sgk' is used to sign the ciphertexts. The function $\tilde{\mathsf{ID}}$ is then modified to return $\perp$ if $(M, r, \mathsf{vk}) = (M_0, r_0, \mathsf{vk}')$ or if $(M, r, \mathsf{vk}) = (M_1, r_1, \mathsf{vk}')$. We omit further details.

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 $(M_0, r_0, 1, \mathsf{sk}')$ and $(M_1, r_1, 1, \mathsf{sk}')$ while answering $\mathcal{A}$'s challenge query. In this case we have $C_s(M_0, r_0, 1, \mathsf{sk}') = C_s(M_1, r_1, 1, \mathsf{sk}') = \mathsf{Decrypt}(s, \mathsf{sk}')$. $\mathcal{B}$ 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 $\tilde{\mathsf{ID}}$ 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 CHES-NMCPA-$\textsc{Game}^{\mathsf{CHE}}_{b,\mathcal{A}}(\lambda, n)$ (b is the random bit selected by $\mathcal{CHE}.\mathcal{C}$ 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 CHES-NMCPA-$\textsc{Game}^{\mathsf{CHE}}_{b,\mathcal{A}}(\lambda, n)$ 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 CHES-NMCPA-$\textsc{Game}^{\mathsf{CHE}}_{\beta,\mathcal{A}}$ is indistinguishable from the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ that is in turn interacting with $\mathcal{FE}.\mathcal{C}$ that sets $b = \beta$. We do so by considering a sequence of hybrid experiments and then showing that adjacent hybrid experiments are indistinguishable.

Hybrid H_0^β

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

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

Hybrid H_1^β Hybrid H_1^β differs from H_0^β 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^β follows.

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

Next we show that, by the pseudorandomness of F, the views of $\mathcal{A}$ in H_0^β and H_1^β 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^β or in H_1^β. This suffices for proving that $H_0^\beta \approx_c H_1^\beta$. Next we describe S.

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

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^β. 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^β.

Hybrid H_2^β Hybrid H_2^β differs from H_1^β 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^β follows.

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

Next we show that, by the IND-CPA security of encryption scheme Enc, the views of $\mathcal{A}$ in H_1^β and H_2^β are indistinguishable, for $\beta = 0, 1$. We do so by constructing an IND-CPA adversary S for Enc that uses $\mathcal{A}$ as a subroutine and interacts with a challenger $\mathcal{C}$ for 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^β; 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^β. Thus, if the two views can be distinguished, S can break the IND-CPA security of Enc. Next we describe S.

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

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

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

Next we show that, by the IND-CPA security of functional encryption scheme FE, the views of $\mathcal{A}$ in H_2^β and H_3^β 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 $\mathcal{FE}.\mathcal{C}$ for FE. S has the property that if $\mathcal{FE}.\mathcal{C}$ 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^β; on the other hand, if $\mathcal{FE}.\mathcal{C}$ answers encryption queries by encrypting m'_1 then $\mathcal{A}$'s view is exactly the same as in H_3^β. 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 $\mathsf{FE.Pk}$ from $\mathcal{FE}.\mathcal{C}$ and sets $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Setup}(1^\lambda)$. Moreover, S randomly picks $r, r' \leftarrow \{0,1\}^\lambda$ and runs $\mathcal{A}$ on input $\mathsf{CHE.Pk} = \mathsf{FE.Pk}$.
2. **Encryption query.** When $\mathcal{A}$ issues an encryption query for messages (M_0, M_1), S sets
$$m_0 = (M_\beta, r, 0, \mathsf{sk}) \text{ and } m_1 = (M_\beta, r', 1, \mathsf{sk}')$$
issues encryption query (m_0, m_1) to $\mathcal{FE}.\mathcal{C}$, obtains 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 $s' = \mathsf{FE.Enc}(\mathsf{FE.Pk}, (C(M_\beta), 0^\lambda, \bot, 0^\lambda); F(r, C))$, $s = \mathsf{Enc}(\mathsf{pk}', s')$ and issues an encryption query to $\mathcal{FE}.\mathcal{C}$ for a token for circuit C_s. Finally, S returns the token $\mathtt{Tok}$ received from $\mathcal{FE}.\mathcal{C}$ 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

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

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

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

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

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

All it is left to show is that the string $\mathtt{Out}$ computed by $\mathcal{B}$ and then fed as input to $\mathcal{D}$ is indistinguishable from the output of CHES-NMCPA-$\text{GAME}_{b,\mathcal{A}}^{\mathsf{CHE}}$. This is necessary since $\mathcal{B}$ does not have access to Msk and uses a token for function $\tilde{\mathsf{ID}}$ instead. However, observe that $\tilde{\mathsf{ID}}$ differs from the decryption function when the plaintext associated with $\mathsf{Ct}^\star$ is of the form $(M_b, r_b, t, \mathsf{sk})$ or $(M_{1-b}, r_{1-b}, t, \mathsf{sk})$ for some t and sk. In the first case, this means that $\mathcal{A}$ has managed to re-randomize

a ciphertext for 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 λ-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 CHES-NMCPA-Game. Therefore, by the hypothesis on $\mathcal{D}$ we can conclude that $\mathcal{B}$ breaks the IND-CPA security of FE.

5 Acknowledgements

Vincenzo Iovino is supported by a FNR CORE grant (no. FNR11299247) of the Luxembourg National Research Fund. Part of this work was done while Vincenzo Iovino was at the University of Warsaw and was supported by the WELCOME/2010-4/2 grant funded within the framework of the EU Innovative Economy Operational Programme.

References

1. S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO (2)*, pages 500–518, 2013.
2. S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In R. Canetti and J. 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.
3. J. Alwen, M. Barbosa, P. Farshim, R. Gennaro, S. D. Gordon, S. Tessaro, and D. A. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In M. 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.
4. M. Barbosa and P. 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.
5. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273, Providence, RI, USA, Mar. 28–30, 2011. Springer, Berlin, Germany.
6. D. Boneh, G. Segev, and B. Waters. Targeted malleability: homomorphic encryption for restricted computations. In *ITCS*, pages 350–366, 2012.
7. E. Boyle, K. Chung, and R. Pass. On extractability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 52–73, 2014.
8. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

9. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (Standard) LWE. In *FOCS*, pages 97–106, 2011.
10. A. De Caro and V. Iovino. On the power of rewinding simulators in functional encryption. *Designs, Codes and Cryptography*, pages 1–27, 2016.
11. A. De Caro, V. Iovino, A. Jain, A. O'Neill, O. Paneth, and G. Persiano. On the achievability of simulation-based security for functional encryption. In R. Canetti and J. A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 519–535. Springer, 2013.
12. D. Dolev, C. Dwork, and M. 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.
13. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.
14. S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Functional encryption without obfuscation. In E. Kushilevitz and T. Malkin, editors, *Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 480–511. Springer, 2016.
15. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.
16. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, Maryland, USA, May 31 – June 2, 2009. ACM Press.
17. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Berlin, Germany.
18. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2012.
19. V. Goyal, A. Jain, V. Koppula, and A. Sahai. Functional encryption for randomized functionalities. Cryptology ePrint Archive, Report 2013/729, 2013. `http://eprint.iacr.org/`.
20. V. Iovino and K. Żebrowski. Simulation-based secure functional encryption in the random oracle model. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 21–39, 2015.
21. M. Naor. On cryptographic assumptions and challenges (invited talk). In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer, Berlin, Germany.
22. R. Pass, A. Shelat, and V. 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.

23. B. Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 678–697, 2015.

A Syntax and Security of Functional Encryption for Circuits

Definition 3 (FE4C: Functional Encryption Scheme for Circuits). *A* Functional Encryption scheme for Circuits *is a tuple* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Eval})$ *of 4 efficient algorithms with the following syntax:*

1. $\mathsf{FE.Setup}(1^\lambda, 1^n)$ *outputs* public *and* master secret *keys* $(\mathsf{Pk}, \mathsf{Msk})$ *for* security parameter λ *and* length parameter n.
2. $\mathsf{FE.KeyGen}(\mathsf{Msk}, C)$*, on input a master secret key* Msk *for length parameter* n *and an* n*-bit input and* n*-bit output circuit* C*, outputs* token $\mathtt{Tok}_C$.
3. $\mathsf{FE.Enc}(\mathsf{Pk}, M)$*, on input public key* Pk *for length parameter* n *and* plaintext $M \in \{0,1\}^n$*, outputs* ciphertext Ct.
4. $\mathsf{FE.Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$ *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* $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$*,* $\mathtt{Tok} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, C)$ *and* $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, M)$*, the probability that* $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}) \neq C(M)$ *is negligible in* λ.

We formalize security for a FE4C FE by means of the following game FE-IndCPA-Game between a challenger $\mathcal{FE}.\mathcal{C}$ and an adversary $\mathcal{A}$ that can issue two types of queries to $\mathcal{FE}.\mathcal{C}$, *encryption queries* and *token queries*. The definition is essentially the one in [5].

FE-IndCPA-Game$^{\mathsf{FE}}_{\mathcal{A}}(\lambda, n)$

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

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

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

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

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

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 λ takes two extra arguments: a *tag* τ from the set T_λ of λ-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 aux and a random tag τ, 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 $\mathcal{FE}.\mathcal{C}$ and and adversary $\mathcal{A}$ that can issue one single encryption query.

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

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

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

Encryption Query. $\mathcal{FE}.\mathcal{C}$, on input plaintext m and auxiliary message aux, picks a random tag τ from the set T_λ of tags of length λ and returns a ciphertext $\mathsf{Ct} = \mathsf{TFE.Enc}(\mathsf{Pk}, m, \mathsf{aux}, \tau)$ with tag τ.

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

1. $\mathsf{Ct}^\star \neq \mathsf{Ct}$;
2. $\mathsf{TFE.Dec}(\mathsf{Msk}, \mathsf{Ct}^\star) = (m', \mathsf{aux}', \tau')$ 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 λ there exists a negligible function μ such that

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

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 $\mathsf{TFE.Enc}$ that encrypts plaintext m and aux, to pick a random pair of $(\mathsf{vk}, \mathsf{sgk})$ of verification and signing key for a secure signature

scheme ($\mathsf{SigKeyGen}, \mathsf{Sign}, \mathsf{Verify}$) and m, aux are encrypted using vk as tag obtaining Ct. Finally, a signature σ_{Ct} of Ct is computed using the signing key sgk and the resulting ciphertext consists of the pair $(\mathsf{Ct}, \sigma_{\mathsf{Ct}})$. Tokens for function C on ciphertext $(\mathsf{Ct}, \sigma_{\mathsf{Ct}})$ first verify σ_{Ct} and, if successful, proceed to compute $C(m, \mathsf{aux})$. 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).

B IND-CPA CHES

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

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

Setup. $\mathcal{CHE}.\mathcal{C}$ computes $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{CHE.Setup}(1^\lambda, 1^n)$, selects a random $b \in \{0,1\}$ and runs $\mathcal{A}$ on input Pk.

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

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

Output of the Game. Let b' be $\mathcal{A}$'s output. Return 1 (meaning that $\mathcal{A}$ has won) iff $b = b'$.

Definition 5. *A* CHES CHE *is* IND-CPA secure *if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that* $\Pr\left[\text{CHES-INDCPA-GAME}^{\mathsf{CHE}}_{\mathcal{A}}(\lambda, n) = 1\right] \leq 1/2 + \mu(\lambda)$.

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.*

B.1 Implications

IND-CPA CHES *from LWE* Noticing that any fully homomorphic encryption scheme [16] is also an IND-CPA CHES, we have that the results of [9,8] prove the following theorem.

Theorem 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 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 $1/2+\mathsf{nneg}(\lambda)$, for some non-negligible function $\mathsf{nneg}(\cdot)$ 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 $C^\star(M_b^1)$ and $C^\star$ is a circuit that satisfies the condition of the definition. Specifically, $C^\star(m) = m + i$ where i is the smallest integer for which $C^\star$ satisfies the condition of the definition (that is $C^\star(M_0^1) \neq C^\star(M_1^1)$ and none of the tokens asked by $\mathcal{A}$ coincides with $C^\star$ on messages M_0^1 and M_1^1). Since the number of token queries is polynomially bounded, the circuit $C^\star$ 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 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 $1/2 + \mathsf{nneg}(\lambda)$ for some non-negligible function $\mathsf{nneg}(\cdot)$. 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 $\{1, \ldots, q\}$, 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 $1/2(S_0 + S_1) \geq 1/2 + \mathsf{nneg}(\lambda)$.

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 μ, 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: $\frac{S_0+S_1+T_0+T_1}{2} \geq 1/2 + \mathsf{nneg}'(\lambda)$ for some non-negligible function nneg'.

C Single-Message vs Multi-Message NM-CPA CHES

Theorem 5. *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

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

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 [22]. Let $q > 1$ be an upper bound on the the number of encryption queries made by $\mathcal{A}$. Consider the game $\text{CHES-NMCPA-}\textsc{Game}^{\mathsf{CHE}}_{J,\mathcal{A}}(\lambda, n)$ indexed by vector $J = (b_1, \ldots, b_q)$ that specifies that the j-th encryption query is answered by encrypting $M^j_{b_j}$. For $j = 0, \ldots, q$, we define vector $J_j = (1, \ldots, 1, 0, 0, \ldots, 0)$ as the vector whose first j components are 1 and the remaining components are 0. We can now run hybrid arguments since $\text{CHES-NMCPA-}\textsc{Game}^{\mathsf{CHE}}_{0,\mathcal{A}}(\lambda, n)$ corresponds to $\text{CHES-NMCPA-}\textsc{Game}^{\mathsf{CHE}}_{J_0,\mathcal{A}}(\lambda, n)$ and

$\text{CHES-NMCPA-}\textsc{Game}^{\mathsf{CHE}}_{1,\mathcal{A}}(\lambda, n)$ corresponds to $\text{CHES-NMCPA-}\textsc{Game}^{\mathsf{CHE}}_{J_q,\mathcal{A}}(\lambda, n)$. Since $\mathcal{D}$ distinguishes

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

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

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

We can therefore use $\mathcal{D}$ along with 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 $j' \in \{0, \ldots, q-1\}$ and forwards to the challenger the j'-th encryption query $(M^{j'}_0, M^{j'}_1)$ received from $\mathcal{A}$, and forwards to $\mathcal{A}$ the corresponding answer received from the challenger. Instead, for all remaining encryption queries (M^i_0, M^i_1), $\mathcal{A}'$ answers on its own by sending an encryption of M^i_1 when $i < j'$ and of M^i_0 when $i > j'$.

Assume $j = j'$. Notice that when the challenger encrypts M^j_0, the above game corresponds to

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

while when the challenger encrypts M^j_1, the above game corresponds to

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

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