Simulation-Based Secure Functional Encryption in the Random Oracle Model*

Vincenzo Iovino[1] Karol Żebrowski[2]

[1] University of Warsaw, vincenzo.iovino@crypto.edu.pl
[2] University of Warsaw, kz277580@students.mimuw.edu.pl

Abstract

One of the main lines of research in functional encryption (FE) has consisted in studying the security notions for FE and their achievability. This study was initiated by [Boneh et al. – TCC'11, O'Neill – ePrint'10] where it was first shown that for FE the indistinguishability-based (IND) security notion is not sufficient in the sense that there are FE schemes that are provably IND-Secure but concretely insecure. For this reason, researchers investigated the achievability of Simulation-based (SIM) security, a stronger notion of security. Unfortunately, the above-mentioned works and others [e.g., Agrawal et al. – CRYPTO'13] have shown strong impossibility results for SIM-Security. One way to overcome these impossibility results was first suggested in the work of Boneh et al. where it was shown how to construct, in the Random Oracle (RO) model, SIM-Secure FE for restricted functionalities and was asked the generalization to more complex functionalities as a challenging problem in the area. Subsequently, [De Caro et al. – CRYPTO'13] proposed a candidate construction of SIM-Secure FE for all circuits in the RO model assuming the existence of an IND-Secure FE scheme for circuits *with RO gates*. This means that the functionality has to depend on the RO, thus it is not fixed in advance as in the standard definitions of FE. Moreover, to our knowledge there are no proposed candidate IND-Secure FE schemes for circuits with RO gates and they seem unlikely to exist. In this paper, we propose the first constructions of SIM-Secure FE schemes in the RO model that overcome the current impossibility results in different settings. We can do that because we resort to the two following models:

- In the public-key setting we assume a bound on the number of queries but this bound only affects the *running-times* of our encryption and decryption procedures. We stress that our FE schemes in this model are SIM-Secure and have ciphertexts and tokens of *constant-size*, whereas in the standard model, the current SIM-Secure FE schemes for general functionalities [De Caro et al., Gorbunov et al. – CRYPTO'12] have ciphertexts and tokens of size growing as the number of queries.
- In the symmetric-key setting we assume a timestamp on both ciphertexts and tokens. This is reasonable because, in the symmetric-key setting, there is only one user that encrypts and generates tokens. In this model, we provide FE schemes with short ciphertexts and tokens that are SIM-Secure against adversaries asking an *unbounded* number of queries.

Both results also assume the RO model, but not functionalities with RO gates and rely on extractability obfuscation w.r.t. distributional auxiliary input [Boyle et al. – TCC'14] (and other standard primitives) secure only in the *standard model*.

Keywords: Functional Encryption, Random Oracle Model, Simulation-Based Security, Obfuscation.

*This work is an extended abstract of the Master's Thesis of the second author and appears in the proceedings of LATINCRYPT 2015.

Contents

1 Introduction **1**

2 Definitions **6**
2.1 Functional Encryption 6
2.2 Multi-Input Functional Encryption 7
2.3 Collision-Resistant Indistinguishability Security for MI-FE 7
2.4 Extractability obfuscation w.r.t. distributional auxiliary input 8

3 Our Transformations **9**
3.1 $(q_1, q_c, \mathsf{poly})$-SIM-Security 9
3.1.1 Trapdoor Machines 9
3.1.2 RO-based Transformation 10
3.2 (q_1, q_c, q_2)-SIM-Security with short tokens 17
3.3 $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-SIM-Security in the Timestamp model 18

4 Constructions of CRIND-Secure MI-FE from $\mathsf{e}\mathcal{O}$ **19**

5 Acknowledgments **23**

A Standard Notions **26**
A.1 Collision-resistant Hash Functions 26
A.2 Symmetric-key encryption 26

B Functional Signature Schemes **26**

C FE and its IND-Security **28**

D MI-FE and its IND-Security **30**

1 Introduction

Functional Encryption (FE) is a sophisticated type of encryption that allows to finely control the amount of information that is revealed by a ciphertext. Unlike in the case of classical cryptosystems, a general study of the security of FE did not appear initially. Instead, progressively more expressive forms of FE were constructed in a series of works (see, e.g., [BDOP04, BW07, KSW08, LOS+10, OT12, Wat12]) that adopted indistinguishability-based (IND) notions of security culminating in the breakthrough of Garg *et al.* [GGH+13]. The study of simulation-based (SIM) notions of security for functional encryption were initiated only recently by Boneh, Sahai, and Waters [BSW11] and O'Neill [O'N10].

Quite interestingly, they show there exists clearly insecure FE schemes for certain functionalities that are nonetheless deemed secure by IND-Security, whereas these schemes do not meet the stronger notion of SIM-Security. For this reason, researchers have started a further theoretical study of FE that includes either negative results showing SIM-Security is not always achievable [BSW11, BO13, AGVW13] or alternative models overcoming the impossibility results [CI13, AAB+13]. On the positive direction, Boneh *et al.* [BSW11] showed the existence of SIM-Secure FE schemes in the Random Oracle (RO, in short) [BR93] model for restricted functionalities (i.e., Attribute-based Encryption), and at the same time they left as a challenging problem the construction of FE for more sophisticated functionalities that satisfy SIM-Security in the RO model.

More recently, De Caro *et al.* [DIJ+13] showed how to overcome all known impossibility results assuming the existence of IND-Secure schemes for circuits with *random oracle gates*. This is a very strong assumption for which we do *not* know any candidate scheme and their existence seems unlikely.[1] Furthermore, their scheme incurs the following theoretical problem. First of all, recall that in a FE system for functionality $F : K \times X \to \Sigma$, defined over *key space* K, *message space* X and *output space* Σ, for every *key* $k \in K$, the owner of the master secret key Msk associated with master public key Pk can generate a secret key $\mathtt{Tok}_k$ that allows the computation of $F(k, x)$ from a ciphertext of x computed under master public key Pk. Thus, in a standard FE scheme the functionality is fixed in advance, and the scheme should allow to compute over encrypted data accordingly to this functionality. Instead, in the scheme for the RO model of De Caro *et al.* [DIJ+13], the functionality does *depend* on the RO, and thus, even their implicit definition of functionality and FE scheme is not standard. Therefore, their scheme is not satisfactory.

This leads to the main question that we study in this work:

> Can we achieve standard FE schemes in the (conventional) Programmable RO model from reasonable assumptions?

Our results answer affirmatively to this question demonstrating the existence of SIM-Secure schemes in the RO model with short parameters. Recall that the impossibility result of [AGVW13] shows that in a SIM-Secure FE scheme, the size of the ciphertexts has to grow as the number of token queries (see also [GVW12a]). Furthermore, De Caro and Iovino [CI13] also showed a similar impossibility result, namely that (for the standard model) in an adaptively SIM-Secure FE scheme, the size of the tokens has to grow as the number of the ciphertext queries. On the other hand, our results also provide schemes for the RO model that are SIM-Secure but in which the size of the tokens and the ciphertexts is constant.[2] Before presenting our positive results in

[1] This issue was first noticed by several researchers [AGK+13] and personally communicated by Jonathan Katz to the authors of the work [DIJ+13].

[2] Specifically, in our main transformation the size of the tokens is constant if we employ a collision-resistant

more detail, we prefer to first sketch our techniques so to highlight the technical problems that led to our constructions and models.

Our techniques. We recall (a simplified version of) the transformation of De Caro *et al.* [DIJ+13] to bootstrap an IND-Secure scheme for Boolean circuits to a SIM-Secure scheme for the same functionality. For sake of simplicity we focus on the non-adaptive setting, specifically on SIM-Security against adversaries asking q non-adaptive queries.[3]

The idea of their transformation is to replace the original circuit with a "trapdoor" one that the simulator can use to program the output in some way. The approach was inspired by the FLS paradigm introduced by Feige, Lapidot and Shamir [FLS90] to obtain zero-knowledge proof systems from witness indistinguishable proof systems. In the transformed scheme, they put additional "slots" in the plaintexts and secret keys that will only be used by the simulator. A plaintext has $2 + 2q$ slots and a secret key will have one. In the plaintext, the first slot is the actual message m and the second slot will be a bit flag indicating whether the ciphertext is in trapdoor mode. and the last $2q$ slots will be q pairs (r_i, z_i), where r_i is a random string and z_i is a programmed string. These $2q$ slots are used to handle q non-adaptive queries. On the other hand, in a secret key for circuit C, the slot is a random string r, that will be equal to one of the r_i in the challenge ciphertext. For evaluation, if the ciphertext is not in trapdoor mode ($\mathsf{flag} = 0$) then the new circuit simply evaluates the original circuit C of the message m. If the ciphertext is in trapdoor mode, if $r = r_i$ for some $i \in [q]$ then the transformed circuit outputs z_i.

A natural approach to shorten the size of the ciphertexts in the RO model would be the following. Recall that a Multi-Input FE (MI-FE) scheme [GGG+14, GKL+13, GGJS13] is a FE over multiple ciphertexts. Let our starting scheme be a MI-FE over 2-inputs. Then, instead of encoding the slots (r_i, z_i)'s in the ciphertext, we could add to the ciphertext a tag $\mathsf{tag_c}$ (that is, the encryption would consist of a ciphertext plus the tag in clear) such that the simulator can program the RO on this point to output the values (r_i, z_i)'s. Now the ciphertext would consist of only a short ciphertext ct and the short tag $\mathsf{tag_c}$. At decryption time, we could "decompress" $\mathsf{tag_c}$ to get some string y, encrypt it with the public-key of the of the MI-FE scheme to produce $\mathsf{ct_2}$ and finally feed $\mathsf{ct_1}$ and $\mathsf{ct_2}$ to the multi-input token. Then, the simulator could program the RO so to output the values (r_i, z_i)'s on the point $\mathsf{tag_c}$. The functionality would be thus modified so that, in trapdoor mode, the output would be taken by the values (r_i, z_i)'s (specifically, choosing the values z_i corresponding to the string r_i in the token). Therefore, any token applied to the simulated ciphertext (that is in trapdoor mode) should decrypt the same output as the real ciphertext would do.

This simple approach incurs more problems, the most serious being that the adversary could feed the multi-input token with a *different* ciphertext that does not correspond to $\mathcal{RO}(\mathsf{tag_c})$, thus detecting whether the first ciphertext is in normal or trapdoor mode. In fact, notice that in normal mode the second ciphertext does not affect the final output, whereas in trapdoor mode, the output only depends on the second ciphertext fed to the multi-input token. Another problem here is that $\mathcal{RO}(\mathsf{tag_c})$ should not contain the values (r_i, z_i)'s in clear, but this is easily solved by letting $\mathcal{RO}(\mathsf{tag_c})$ be an encryption of them and putting the corresponding secret-key

hash function of variable-length, otherwise their size only depends on the encoding of the value and thus can be sub-logarithmic. Similarly, for the timestamp model of Section 3.3, both tokens and ciphertexts need to encode a temporal index that being a number at most equal to the number of queries issued by any PPT adversary, will be at most super-logarithmic, and thus can be encoded with a string of poly-logarithmic size. For simplicity, henceforth, we will claim that our constructions have tokens of constant size omitting to specify this detail.

[3]Henceforth, we mean by non-adaptive queries the queries that the adversary asks before seeing the challenge ciphertext and adaptive queries the queries the adversary asks after seeing it.

in the first ciphertext. So, the main technical problem is:

> How can we force the adversary to feed the 2-inputs token with a second ciphertext that encrypts $\mathcal{RO}(\mathsf{tag_c})$?

Note that in the case of FE schemes that support functionalities with RO gates, this can be easily done by defining a new functionality that first tests whether the second input equals $\mathcal{RO}(\mathsf{tag_c})$, but in the "pure" RO model this solution can not be applied. Our patch is to add a new slot h of *short* size in the first ciphertext. Such value h is set to the hash of $\mathcal{RO}(\mathsf{tag_c})$ with respect to a Collision-Resistant Hash Function (CRHF) Hash, i.e., $h = \mathsf{Hash}(\mathcal{RO}(\mathsf{tag_c}))$. Furthermore, we modify the transformed functionality so that it first checks whether $\mathsf{Hash}(\mathcal{RO}(\mathsf{tag_c})) = h$. If this test fails, the functionality outputs an error $\perp$.

The intuition is that with this modification, the adversary is now forced to use a second ciphertext that encrypts $\mathcal{RO}(\mathsf{tag_c})$ since otherwise it gets $\perp$ on both real or simulated ciphertext, and so, under the IND-Security of the MI-FE scheme, it seems that the adversary can not tell apart a real ciphertext from a simulated ciphertext. Unfortunately, we are not able to prove the security of this transformation assuming only the standard notion of IND-Security for MI-FE. In fact, notice that there *exist* second inputs for the modified functionality that distinguish whether the first input has the flag set to normal or trapdoor mode, namely inputs that correspond to collisions of Hash with respect to h and $\mathcal{RO}(\mathsf{tag_c})$. That is, any another second ciphertext that encrypt a value $y \neq \mathcal{RO}(\mathsf{tag_c})$ such that $\mathsf{Hash}(y) = h$ allows to distinguish whether the first ciphertext is in normal or trapdoor mode. Furthermore, it is not possible to make direct use of the security of the CRHF. The problem is that the definition of (2-inputs) MI-FE is too strong in that it requests the adversary to output two challenge message pairs (x_0, y) and (x_1, y) such that for any function f for which the adversary asked a query $f(x_0, \cdot) = f(x_1, \cdot)$. In our case, this does not hold: there exists a set of collisions C such that for any $z \in C$, $\mathsf{Hash}(z) = h$ and $f(x_0, z) \neq f(x_1, z)$. However, notice that it seems difficult for the adversary to find such collisions. Moreover, due to the requirement that the ciphertext must have short size, we are unable to use standard techniques already employed in literature.

Our assumptions and CRIND-Security. For these reasons, we need to extend the notion of MI-FE to what we call *collision-resistant indistinguishability* (CRIND, in short)[4]. In Section 4 we provide an instantiation of this primitive from extractability obfuscation w.r.t. distributional auxiliary input [BCP14] (cf. Remark 2.9). We think that this definition can be of independent interest since it is more tailored for the applicability of MI-FE to other primitives. We are aware that other possibilities to overcome our problem would be either to directly modify our transformation or to modify existing construction of MI-FE to satisfy this property, and thus avoiding the repeated use of obfuscation and NIZKs that we do. Nevertheless, we prefer to follow a modular approach to the aim of obtaining clear and simple constructions. Similar considerations also hold for the construction of CRIND-Secure MI-FE schemes and we leave to future research the generalization of this primitive and its construction from weaker assumptions. The reader may have noticed that the security of the second ciphertext guaranteed by the underlying MI-FE is not *necessary*. That is, our transformation would work even assuming 2-inputs MI-FE systems that take the second input *in clear*. In fact, CRIND-Security does *not* imply IND-Security for MI-FE schemes but this suffices for our scopes.

[4]Maybe, a better name would have been "differing-inputs indistinguishability" but we do not adopt this name to not overlap with differing-inputs obfuscation and because it recalls the reason to advocate this stronger notion for our transformations.

Roughly speaking, in CRIND-Security the security is quantified only with respect to *valid* adversaries[5], where an adversary is considered valid if it only submits challenges m_0, m_1 and asks a set of queries K that satisfy some "hardness" property called *collision-resistance compatibility*, namely that it is difficult to find a second input m_2 such that $F(k, m_0, m_2) \neq F(k, m_1, m_2)$ for some $k \in K$. Since in the reductions to CRIND-Security it is not *generally* possible to directly check the hardness of (K, m_0, m_1), the definition dictates (1) the existence of an efficient *checker* algorithm that approves *only* (but possibly not all) valid triples (K, m_0, m_1) (i.e., the checker can detect efficiently if a triple is collision-resistant compatible) and (2) that an adversary is valid if it only asks triples approved by the checker. We defer the details of the definition to Section 2.3. Next, in a security reduction to CRIND-Security (i.e., when we need to prove the indistinguishability of two hybrid experiments assuming the CRIND-Security), the main task is to define an appropriate checker and prove that triples that are not collision-resistant compatible are rejected by it. This is usually done by checking that messages and keys satisfy an appropriate format. For instance, in the above case, the checker will check whether the machine (corresponding to the token) uses as sub-routine the specified CRHF and that the challenge messages and such machine have the right format.

The construction of CRIND-Secure schemes follows the lines of the construction of fully IND-Secure FE schemes of Boyle *et al.* Namely, the encryption of the first input m_1 will be an obfuscation of a machine that has embedded m_1 and a verification key for a signature scheme and takes as input a signature of a machine M and a second input m_2 and (1) checks the validity of the signature and (2) if such test passes outputs $M(m_1, m_2)$. For the same resonas of Boyle *et al.* we need to resort to functional signatures. Details along with a broader overview can be found in Section 4.

The above presentation is an oversimplification that skips some minor details and the modifications necessary to handle adaptive token queries and many ciphertext queries. We defer the reader to the Sections 3.1 and 3.2 for more details.

Our models and results. The reader may have noticed that the output of $\mathcal{RO}$ has to be "big", i.e., its size depends on the number of queries. Of course, we could assume that its range has constant size and replace a single invocation of the RO with range of size $> q$ with many invocation of a RO with range of constant size, but also in this case the *running-time* of the encryption and decryption procedures would have to depend on the number of queries. Here, it is the novelty of our approach. All the parameters of our SIM-Secure public-key FE scheme (including ciphertexts and tokens) have *constant* size but the cost of the "expansion" is moved from the length of ciphertexts and tokens to the running-time of the encryption and decryption procedures. That is, our SIM-Secure public-key FE scheme stills depends on q in the setup and running-time, but the *size* of the ciphertexts and tokens is *constant*. The results we achieve can be summarized as follows:

- $(q_1, q_c, \mathsf{poly})$-SIM-Security with ciphertext of constant size and tokens of size q_c. That is, SIM-Security against adversaries asking bounded non-adaptive token queries, bounded ciphertext queries, and unbounded adaptive token queries. In this case the size of the ciphertexts is constant but the size of the tokens grows as the number of ciphertext queries (and thus is constant in the case of 1 ciphertext query). This is known to be impossible in the standard model due to the impossibility of Agrawal *et al.* [AGVW13] (precisely this impossibility does not rule out the existence of schemes that satisfy this notion of security but it rules out the existence of schemes that satisfy both this notion of security *and* have

[5]We can recast IND-Security in a similar way by defining valid adversaries that only ask queries and challenges satisfying the compatibility property.

short ciphertexts). Moreover, in this case the encryption and decryption procedures have running-times depending on q_1.

- (q_1, q_c, q_2)-SIM-Security with both ciphertexts and tokens of constant size. That is, SIM-Security against adversaries asking bounded token (both non-adaptive and adaptive) and ciphertext queries but with both ciphertext and token of constant size. In the standard model this is known to be impossible due to the impossibility result of De Caro and Iovino [CI13] for SIM-Security against adversaries asking unbounded ciphertext queries and bounded adaptive token queries (this impossibility is essentially an adaptation of the impossibility of Agrawal *et al.* [AGVW13]). In this case, the encryption and decryption procedures have running-times depending on $\max\{q_1, q_c, q_2\}$.

- We show how to remove the afore-mentioned limitation in a variant of the symmetric-key model where ciphertexts and tokens are tagged with a timestamp that imposes an order on their generation (i.e., the i-th token/ciphertext generated in the system is tagged with the value i). This model is reasonable because in the symmetric-key setting, the user that set-up the system is the same entity that generates tokens and ciphertexts as well.[6] Moreover, most of the notable applications of FE are for the symmetric-key setting (e.g., to cloud computing, where a client delegates her encrypted data to a server and later can send tokens to the server to compute specific functions of her choice). We defer the reader to Section 3.3 for more details.

In the above presentation we skipped another technical issue that our approach faces. Specifically, in the Boolean circuit model, the token size would be at least as big as the total size of the bits encrypted in all ciphertext, thus of size dependent of q. Notice that this would already be an improvement with respect to the known bounded FE schemes for Boolean circuits of Gorbunov *et al.* [GVW12b]. However, for the sake of providing constructions with optimal parameters we work in the *Turing Machine* model of computation, though for all our results except that of Section 3.3 it is possible to provide slightly different constructions in the circuit model that improve the current results for the standard model.[7]

The optimality and the soundness of our results. It is easy to see that SIM-Security in the standard model but for schemes with procedures of running-time dependent on the number of queries is impossible as well. Moreover, we think that SIM-Security in the RO model with a constant number of RO calls is impossible to achieve as well, though we were not able to prove an impossibility result for it and leave to future work to set positive or negative results. Anyway, one could object that if we instantiate the RO with any concrete hash function, the resulting scheme is *not* "SIM-Secure" due to the impossibility results. This problem is also shared with the constructions for the RO model of De Caro *et al.* [DIJ+13] and Boneh *et al.* [BSW11]. What does it mean?

What the impossibility results say is that there are adversaries for which there exists no simulator though we do not know any concrete attacks on these schemes. This is different from the counter-examples of Canetti *et al.* [CGH98] where they were presented signature schemes provably secure in the RO model but *concretely* insecure when instantiated with *any* hash function. In our view, this could merely mean that general definitions of SIM-Security are

[6] The same considerations also hold in applications where many users share the same secret-key. Indeed, in this case one needs to assume that such users trust each other, and thus they will tag the ciphertexts and tokens with the correct timestamp.

[7] The main focus of the work of Goldwasser *et al.* [GGJS13] is for the circuit model but they sketch how to extend it to the Turing Machine model. Similar considerations hold for the schemes of Gordon *et al.* [GKL+13]. Further details will be given in the Master's Thesis of the second author.

too strong. Along this direction, the works of De Caro and Iovino [CI13] and Agrawal *et al.* [AAB+13] provide another way to overcome this limitation.

2 Definitions

A *negligible* function $\mathsf{negl}(k)$ is a function that is smaller than the inverse of any polynomial in k. 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 $\{1, \ldots, q\}$. 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 representing their delimited concatenation.

Building blocks. In Appendix B we recall the notion of functional signature schemes; in Appendix A.1 the notion of collision-resistant hash function; and in Appendix A.2 the notion of symmetric-key encryption.

2.1 Functional Encryption

Functional encryption schemes are encryption schemes for which the owner of the master secret can compute restricted keys, called *tokens*, that allow to compute a *functionality* on the plaintext associated with a ciphertext. We start by defining the notion of a functionality.

Definition 2.1 [Functionality] A *functionality* F is a function $F : K \times M \to \Sigma$ where K is the *key space*, M is the *message space* and Σ is the *output space*.

In this work, our FE schemes are for the following functionality[8].

Definition 2.2 [p-TM Functionality] [9] The p-TM functionality for polynomial $p()$ has key space K equals to the set of all Turing machines M, which satisfy the condition: the running time of M on any input m is exactly $p(|m|)$ (depends only on the input length). The message space M is the set $\{0,1\}^*$. For $M \in K$ and $m \in M$, we have p-TM(M,m)= M(m), In this work, for simplicity we assume that the output space is $\{0,1\}$.

Remark 2.3 In case of a scheme with input-specific run time (cf. Definition C.1), we also require that the functionality outputs the run time of machine M on m along with the output of the computation $M(m)$. We will only use schemes with input specific run time in Section 3.3. For the other applications bounded time is sufficient.

The definition of a FE scheme can be found in Appendix C. In Appendix C we recall the standard definition of indistinguishability-based and simulation-based security for functional encryption.

[8] As said in the introduction, it is possible to resort to the circuit model at cost of having slightly different constructions and tokens of longer size.

[9] This functionality was only implicitly assumed in Goldwasser *et al.* [GKP+13] and other works but not formally defined.

2.2 Multi-Input Functional Encryption

Multi-input functional encryption is analogous to functional encryption except that the functionality takes multiple inputs as argument. We recall the definition of multi-input functionalities and MI-FE in Appendix D. In Appendix D we recall the standard definition of indistinguishability-based security for multi-input functional encryption. In particular we would like to draw the attention of the reader on the definition D.2.

Remark 2.4 In this work we do not need to assume IND-Secure MI-FE schemes since we will construct CRIND-Secure MI-FE that suffice for our scopes. Specifically, for our purposes the underlying CRIND-Secure MI-FE scheme that we use could have an encryption procedure that encrypts the second input outputting it in *clear*. Alternatively, we could have defined a new primitive with its security. Instead, to avoid to introduce new syntax, we construct and use schemes that formally follow the syntax of MI-FE (with its correctness) but assuming a *different and incomparable* security notion for them.

2.3 Collision-Resistant Indistinguishability Security for MI-FE

As we mentioned in the construction overview sketched in Section 1, we need a different notion of MI-FE security. Here, we consider only the 2-inputs case, since this is suited for our main transformation but it is straightforward to extend it to the n-ary case.

Furthermore, in Section 4 we will show how to construct a CRIND-Secure MI-FE scheme from extractability obfuscation w.r.t. distributional auxiliary input [BCP14] (cf. Remark 2.9). We presented an informal discussion of the definition in Section 1. We now present the formal definition.

The collision-resistant indistinguishability-based notion of security for a multi-input functional encryption scheme $\mathsf{MI\text{-}FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, M) is formalized by means of the following experiment $\mathsf{CRIND}^{\mathsf{MI\text{-}FE}}_{\mathcal{A}}$ with an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. Below, we present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

$\mathsf{CRIND}^{\mathsf{MI\text{-}FE}}_{\mathcal{A}}(1^\lambda)$

1. $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{MI\text{-}FE.Setup}(1^\lambda)$;
2. $r \xleftarrow{R} \{0,1\}^\lambda$;
3. $(x_0, x_1, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk}, r)$ where we require that $|x_0| = |x_1|$;
4. $b \xleftarrow{R} \{0,1\}$;
5. $\mathsf{Ct}_1 \leftarrow \mathsf{MI\text{-}FE.Encrypt}(\mathsf{Ek}_1, (x_b||r))$;
6. $b' \leftarrow \mathcal{A}_1^{\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk}, \mathsf{Ct}_1, \mathtt{st})$;
7. **Output:** $(b = b')$.

We make the following additional requirements:

- Collision-resistance compatibility. Let K denote a set of keys. We say that a pair of messages x_0 and x_1 is *collision-resistant compatible* with K if it holds that: any (possibly, non-uniform) PPT algorithm $\mathcal{B}$ given the security parameter 1^λ and (r, x_0, x_1) for r uniformly

distributed in $\{0,1\}^\lambda$, can find y satisfying inequality $F(k,(x_0||r),y) \neq F(k,(x_1||r),y)$ for some $k \in K$ with at most negligible (in λ) probability, where probability is taken over $r \xleftarrow{R} \{0,1\}^\lambda$ and the random coins of $\mathcal{B}$.[10]

- Efficient checkability. We assume that there exist efficient checker algorithm $\mathsf{Checker}$, which takes as input (k, x_0, x_1) and outputs false if x_0 and x_1 are *not* collision-resistant compatible with $\{k\}$ (i.e., the singleton set containing the key k).
- Validity. We say that an adversary $\mathcal{A}$ in the above game is *valid* with respect to a checker $\mathsf{Checker}$ with efficient checkability if during the execution of the above game, $\mathcal{A}$ outputs challenge messages x_0 and x_1 of the same length and asks a set of queries K in the key space of the functionality such that for any $k \in K$, $\mathsf{Checker}(k, x_0, x_1) = \mathsf{true}$ (i.e., the adversary only asks queries and challenges approved by the checker).

The advantage of a valid adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MI\text{-}FE,IND}}(1^\lambda) = |\mathrm{Prob}[\mathsf{CRIND}_{\mathcal{A}}^{\mathsf{MI\text{-}FE}}(1^\lambda) = 1] - 1/2|.$$

Definition 2.5 We say that a 2-inputs MI-FE scheme is *collision-resistant indistinguishably secure* (CRIND-Secure, in short) if for any checker $\mathsf{Checker}$ satisfying efficient checkability, it holds that all PPT adversaries $\mathcal{A}$ *valid* with respect to $\mathsf{Checker}$ have at most negligible advantage in the above game.

Remark 2.6 It can appear that the existence of this $\mathsf{Checker}$ is unlikely since it seems not possible to check the condition directly. Anyway, for large classes of functionalities this is possibly efficient. For instance, in our case the trapdoor machines have a specified format that includes a sub-routine for the CRHF Hash. The checker must only be able to check that this subroutine equal the code of Hash and that the trapdoor machine and the messages have the right format. For machines and messages that satisfy these checks, the condition of collision-resistance compatibility is guaranteed. There is no computational property to be checked here, but only a syntactical one.

Remark 2.7 We stress that CRIND-Security does not imply IND-Security for MI-FE but we do not need this. In fact, notice that for our applications the second input could be in clear. We also point out that the notion of CRIND-Security could be generalized so to have as special case IND-Security but to not overburden the presentation we do not do this and defer it to the Master's Thesis of the second author.

2.4 Extractability obfuscation w.r.t. distributional auxiliary input

Boyl *et al.* [BCP14] defined obfuscators secure against general distributional auxiliary input. We recall their definition (cf. Remark 2.9).

Definition 2.8 A uniform PPT machine $\mathsf{e}\mathcal{O}$ is an *extractability obfuscator w.r.t. (general) distributional auiliary input* for the class of Turing Machines $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following properties:

[10] It can appear that the definition be not well-defined because we do not specify how the key k is related to the security parameter. To understand this, you may imagine that k be the code of some algorithm P (ant thus of constant-size) to compute a keyed hash function $\mathsf{Hash}(\cdot,\cdot)$. The program P takes an hashing key s computed with respect to an arbitrarily long security parameter λ and an input x and computes $\mathsf{Hash}(s,x)$. Therefore in the above definition, k (along with the functionality F) plays the role of P and thus can have constant size whereas r plays the role of the hashing key s that depends instead on the security parameter.

- (Correctness): For all security parameter λ, all $M \in \mathcal{M}$, all inputs x, we have

$$\Pr\left[M' \leftarrow \mathsf{e}\mathcal{O}(1^\lambda, M) \colon M'(x) = M(x) \right] = 1.$$

- (Polynomial slowdown): There exist a universal polynomial p such that for any machine M, we have $|M'| \leq p(|M|)$ for all $M' = \mathsf{e}\mathcal{O}(1^\lambda, M)$ under all random coins.

- (Security): For every non-uniform PPT adversary $\mathcal{A}$, any polynomial $p(\lambda)$, and efficiently sampleable distribution $\mathcal{D}$ over $\mathcal{M}_\lambda \times M_\lambda \times \{0,1\}^\star$, there exists a non-uniform PPT extractor E and polynomial $q(\lambda)$ and negligible function $\mathsf{negl}(\lambda)$ such that, for every $\lambda \in \mathbb{N}$, with probability $\geq 1 - \mathsf{negl}(\lambda)$ over $(M_0, M_1, z) \leftarrow \mathcal{D}(1^\lambda)$, it holds that:
If $\Pr\left[b \xleftarrow{R} \{0,1\}; M' \leftarrow \mathsf{e}\mathcal{O}(1^\lambda, M_b) : \mathcal{A}(1^\lambda, M', M_0, M_1, z) = b \right] \geq \frac{1}{2} + \frac{1}{p(\lambda)}$,
then $\Pr\left[w \leftarrow \mathsf{E}(1^\lambda, M_0, M_1, z) : M_0(w) \neq M_1(w) \right] \geq \frac{1}{q(\lambda)}$

Note that in the above definition the extractor E may depend on the the adversary A and the distribution $\mathcal{D}$.

Remark 2.9 In light of the recent "implausibility" results on extractability obfuscation with auxiliary input [GGHW13, BP13], we would like to point out that our results are based on *specific* distributions for which no implausibility result is known. Same considerations were also made in [BCP14].

3 Our Transformations

3.1 $(q_1, q_c, \mathsf{poly})$-SIM-Security

In this section, we show that assuming a CRIND-Secure (in the standard model) MI-FE scheme for p-TM_2 (2-inputs Turing machines with run time equal to a polynomial p) for any polynomial p, it is possible to construct a SIM-Secure functional encryption scheme in the RO model for functionality p-TM for any polynomial p. Moreover, this is possible also for FE schemes with input-specific run time. The resulting scheme is secure for a bounded number of messages and non-adaptive token queries, and unbounded number of adaptive key queries. Moreover, it enjoys ciphertexts and tokens of size not growing with the number of non-adaptive queries, overcoming the impossibility result of Agrawal *et al.* [AGVW13] for the standard model. In Section 4 we will show how to construct a CRIND-Secure MI-FE from extractability obfuscation w.r.t. distributional auxiliary input [BCP14] (cf. Remark 2.9).

3.1.1 Trapdoor Machines

The idea of our transformations is to replace the original machine with a "trapdoor" one that the simulator can use to program the output in some way. This approach is inspired by the FLS paradigm introduced by Feige, Lapidot and Shamir [FLS90] to obtain zero-knowledge proof systems from witness indistinguishable proof systems. Below we present the construction of trapdoor machine, which works in standard model.

Definition 3.1 [Trapdoor Machine] Fix $q > 0$. Let M be a Turing machine with one input. Let $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ be a symmetric-key encryption scheme with key-space $\{0,1\}^\lambda$, message-space $\{0,1\}^{\lambda+1}$, and ciphertext-space $\{0,1\}^\nu$. We require for simplicity that SE has pseudo-random ciphertexts (see Appendix A.2) and can encrypt messages of variable length (at most

$\lambda + 1$). Let $\mathsf{Hash}\colon \{0,1\}^\lambda \times \{0,1\}^{q\cdot\nu} \to \{0,1\}^\lambda$ be a collision resistant hash function [11]. For $\mathsf{tag_k} = (id_k, c) \in \{0,1\}^{\lambda+\nu}$ define the corresponding *trapdoor machine* $\mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag_k}}$ on two inputs as follows:

Machine $\mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag_k}}(m', R)$
  $(m, \mathsf{flag}, sk, h, k_\mathsf{H}) \leftarrow m'$
  If $\mathsf{Hash}(k_\mathsf{H}, R) \neq h$ then return $\perp$
  If $\mathsf{flag} = 0$ then return $M(m)$
  $(id_k, c) \leftarrow \mathsf{tag_k}$
  $(R_1, \ldots, R_q) \leftarrow R$
  For $i = 1, \ldots, q$ do
    $(id_k{}', v) \leftarrow \mathsf{SE.Dec}(sk, R_i)$
    If $id_k{}' = id_k$ then return v
  return $\mathsf{SE.Dec}(sk, c)$

3.1.2 RO-based Transformation

Overview. In Section 1 we sketched a simplified version of our transformation. Here, we present an overview with more details. The idea is to put additional "slots" in the plaintexts and secret keys that will only be used by the simulator. A plaintext contains five slots and a secret key contains two slots. In the plaintext, the first slot is the actual message m. The second slot is a bit flag indicating whether the ciphertext is in trapdoor mode. The third slot is a random key sk used by SE scheme, the fourth slot is a hash h of $\mathcal{RO}(\mathsf{tag_c})$ (computed with respect to a CRHF) attached to the ciphertext and finally the fifth slot contains a hash function key k_H.

In the secret key, the first slot encodes the actual machine M and the second slot is a random tag $\mathsf{tag_k} = (id_k, c)$. Slot id_k is used by simulator to identify pre-challenge tokens and c is used to convey programmed output value in post-challenge tokens. For evaluation, if the ciphertext is not in trapdoor mode (i.e., $\mathsf{flag} = 0$) then the functionality simply evaluates the original machine M of the message m. If the ciphertext is in trapdoor mode, depending on the nature of the secret key (non-adaptive or adaptive), for $\mathsf{tag_k} = (id_k, c)$, if for some $i \in [q]$, $(id_k, v) = \mathsf{SE.Dec}(sk, R_i)$, then the functionality outputs v, otherwise it outputs $\mathsf{SE.Dec}(sk, c)$. Here R_i is the i-th element in the string R that the machine takes as second input, and is set by the (honest) evaluation procedure to $\mathcal{RO}(\mathsf{tag_c})$.

For sake of simplicity we assume that TM functionality, for which our scheme is constructed, has output space $\{0,1\}$. The construction can be easily extended to work for any TM functionality with bounded output length.

Definition 3.2 [RO-Based Transformation] Let $p(\cdot)$ be any polynomial. Let $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ be a symmetric-key encryption scheme with key-space $\{0,1\}^\lambda$, message-space $\{0,1\}^{\lambda+1}$, and ciphertext-space $\{0,1\}^\nu$. We require for simplicity that SE has pseudo-random ciphertexts (see Appendix A.2) and can encrypt messages of variable length (at most $\lambda+1$). Let $\mathsf{Hash}\colon \{0,1\}^\lambda \times \{0,1\}^{q\cdot\nu} \to \{0,1\}^\lambda$ be a collision-resistant hash function (not modeled as a random oracle). Assuming that the running time of machine M equals exactly $p(|m|)$ on input m, the running time of trapdoor machine $\mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag_k}}$ is bounded by some polynomial $p'(\cdot)$ (cf. Remarks

[11] For sake of simplicity as Hash key we will use a random string of length λ, instead of key generated by Gen. Alternatively, we could feed the Gen algorithm with this randomness.

3.3 and 3.5) Let $\mathsf{MI\text{-}FE} = (\mathsf{MI\text{-}FE.Setup}, \mathsf{MI\text{-}FE.Enc}, \mathsf{MI\text{-}FE.KeyGen}, \mathsf{MI\text{-}FE.Eval})$ be a multi-input functional encryption scheme for the functionality p'-TM_2.

In our construction we assume that the output length of the programmable random oracle $\mathcal{RO}$ equals $q \cdot \nu$ (cf. Remark 3.6).

We define a new (single-input) functional encryption scheme $\mathsf{SimFE}[\mathsf{Hash}, \mathsf{SE}] = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality p-TM as follows.

- $\mathsf{Setup}(1^\lambda)$: runs $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{MI\text{-}FE.Setup}(1^\lambda)$ and chooses random $r \xleftarrow{R} \{0,1\}^\lambda$ and returns a pair (Mpk, r) as public key Pk and Msk as master secret key.

- $\mathsf{Enc}(\mathsf{Pk}, m)$: on input $\mathsf{Pk} = (\mathsf{Ek}_1, \mathsf{Ek}_2, r)$ and $m \in \{0,1\}^*$, the algorithm chooses random $sk \xleftarrow{R} \{0,1\}^\lambda$, $\mathsf{tag_c} \xleftarrow{R} \{0,1\}^\lambda$ and sets $k_\mathsf{H} = r$, then it takes $m' = (m, 0, sk, \mathsf{Hash}(k_\mathsf{H}, \mathcal{RO}(\mathsf{tag_c}), k_\mathsf{H})$, computes $c \leftarrow \mathsf{MI\text{-}FE.Enc}(\mathsf{Ek}_1, m')$ and returns a pair $(c, \mathsf{tag_c})$ as its own output.

- $\mathsf{KeyGen}(\mathsf{Msk}, M)$: on input Msk and a machine M, the algorithm chooses random $id_k \xleftarrow{R} \{0,1\}^\lambda$, $c \xleftarrow{R} \{0,1\}^\nu$and returns $(\mathtt{Tok}, \mathsf{tag_k})$ where $\mathtt{Tok} \leftarrow \mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag_k}})$ and $\mathsf{tag_k} = (id_k, c)$.

- $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$: on input $\mathsf{Pk} = (\mathsf{Ek}_1, \mathsf{Ek}_2, r)$, $\mathsf{Ct} = (c, \mathsf{tag_c})$ and $\mathtt{Tok} = (\mathtt{Tok}', \mathsf{tag_k})$, computes $c' = \mathsf{MI\text{-}FE.Enc}(\mathsf{Ek}_2, \mathcal{RO}(\mathsf{tag_c}))$ and returns the output $\mathsf{MI\text{-}FE.Eval}(\mathtt{Tok}', c, c')$.

Remark 3.3 Efficiency and extensions. The above scheme has ciphertexts of constant size and token of constant size and as we will show it satisfies $(q_1, 1, \mathsf{poly})$-SIM-Security. It can be upgraded to support $(q_1, q_c, \mathsf{poly})$-SIM-Security at the cost of having tokens of size depending on q_c. As in De Caro *et al.* [DIJ+13], this can be done by replacing c in the token with q_c values to program the answers to the q_c challenge ciphertexts but we omit details. We remark that the bound q_1 does not affect the value p because p is polynomial in the length of the input messages and not on the security parameter and this fact rules out the problem that p' would have to be set to a value $> \max\{q_1, q_c\}$ to allow the machine to read the inputs. Moreover, the transformation would need to take as input such bounds q_1, q_c (cf. Remark C.7) but for simplicity we omitted such details.

Remark 3.4 If the underlying scheme has input specific run time (cf. Definition C.1) then we are able to provide a modified construction, with input-specific run time as well. Note that in this case the simulator receives the run time of any queried machine on the challenge message (cf. Remark 2.3), since the functionality returns it along with the output of the computation. Therefore the simulator can program not only the desired output value, but could also eventually extend the run time of the trapdoor machine.

Remark 3.5 Actually we should use a machine that has running time depending only on input lengths (cf. Definition D.2), expressed as a polynomial. Therefore, in practice instead of $M' = \mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag_k}}$ we would take a machine that runs M' and then extends the running-time to the desired value $p'(\cdot)$. To not overburden the presentation we omit this detail.

Remark 3.6 Alternatively, instead of using and program $\mathcal{RO}$ at one input $\mathsf{tag_c}$ and having output proportionally long to q, we could program it at q inputs $(\mathsf{tag_c}, 1), \ldots, (\mathsf{tag_c}, q)$. This way we could avoid the correlation between q and $\mathcal{RO}$ output length. However, for simplicity we resort to the first option.

Theorem 3.7 Suppose MI-FE is CRIND-Secure in the standard model. Then SimFE is $(q, 1, \mathsf{poly})$-SIM-Secure (cf. Remark C.5) in the random oracle model. Furthermore, this can be extended to $(q_1, q_c, \mathsf{poly})$-SIM-Security as discussed in remark 3.3 and if MI-FE satisfies the properties of succinctness and input-specific time, so SimFE does.

Security proof overview. We conduct the security proof of our construction by a standard hybrid argument. To move from real world experiment to ideal one we use the following hybrid experiments:

- The first hybrid experiment corresponds to the real experiment.
- The second hybrid experiment is identical to the previous one except that the random oracle is programmed at point $\mathsf{tag_c}$ so to output the encryption of the desired output values on pre-challenge queries, and post-challenge queries are answered with tokens that have embedded appropriate encrypted output values. Moreover, *all* these values are encrypted using the underlying SE scheme with a *randomly chosen* secret-key sk'. Notice that in this experiment the secret-key sk' is *uncorrelated* to the secret-key sk embedded in ciphertext.
- The third hybrid experiment is identical to the previous one except that the ciphertext contains the same secret-key sk' used to program the RO.
- In last step we switch the flag slot in the ciphertext to 1 indicating the trapdoor mode. At the same time we change the content of message slot m to $0^{|m|}$. This is necessary due to the fact that simulator only knows the challenge message length, but not the message itself.

One can reduce the security of first two transitions to the ciphertext pseudo-randomness of SE scheme and to the CRIND-Security of underlying MI-FE scheme. The proof in these cases is pretty straightforward.

One could be tempted to reduce the indistinguishability security of last two hybrids to both collision resistance of used hash function and IND-Security on MI-FE. However, the security reduction is not obvious. The adversary could recognize the simulation by finding a string R different from $\mathcal{RO}(\mathsf{tag_c})$ for which $\mathsf{Hash}(k_\mathsf{H}, R) = \mathsf{Hash}(k_\mathsf{H}, \mathcal{RO}(\mathsf{tag_c}))$, and applying the evaluation algorithm to this value as second input. The output of evaluation algorithm in this case would be different than expected. Although the adversary would contradict the collision resistance of Hash, we are not able to construct algorithm based on that adversary, which breaks the hash function security.

Therefore we need to rely on the CRIND-Security of MI-FE. Moreover, for completeness we will only assume CRIND-Security and never IND-Security.

Proof: Suppose that there is an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against SimFE that outputs at most 1 message, $q = q(\lambda)$ pre-challenge and $p = p(\lambda)$ post-challenge token queries. Note that here p is an unbounded polynomial, not fixed a priori, but q is fixed a priori. We construct a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ as follows (note that simulator can program the random oracle $\mathcal{RO}$).

- Let m be the challenge message output by $\mathcal{A}_0$.

  Sim_0 receives as input the message length $|m|$, the public parameter $\mathsf{Pk} = (\mathsf{Ek}_1, \mathsf{Ek}_2, r)$, the q non-adaptive key queries $M_1, \ldots, M_q$ made by $\mathcal{A}_0$, along with the values $z_1 = M_1(m), \ldots, z_q = M_q(m)$ and the tokens $(\mathtt{Tok}_1, \mathsf{tag}_{\mathsf{k}1}), \ldots, (\mathtt{Tok}_q, \mathsf{tag}_{\mathsf{k}q})$ generated by KeyGen

to answer $\mathcal{A}_0$'s non-adaptive token queries, where $\mathsf{tag}_{\mathsf{k}i} = (id_{ki}, c_i)$. Sim_0 proceeds as follows.

The simulator chooses random $sk \xleftarrow{R} \{0,1\}^\lambda$ and sets $k_{\mathsf{H}} = r$, and stores them in the state $\mathtt{st}'$.

For each $1 \leq i \leq q$, the simulator computes $R_i = \mathsf{SE.Enc}(sk, (id_{ki}, z_i))$ and concatenates the encryptions as $R = R_1 || \ldots || R_q$.

The simulator chooses random $\mathsf{tag}_{\mathsf{c}} \in \{0,1\}^\lambda$ and programs random oracle output by setting $\mathcal{RO}(\mathsf{tag}_{\mathsf{c}}) = R$.

Sim_0 computes $\mathsf{Ct} \leftarrow \mathsf{MI\text{-}FE.Enc}(\mathsf{Ek}_1, (0^{|m|}, 1, sk, \mathsf{Hash}(k_{\mathsf{H}}, R), k_{\mathsf{H}}))$ and outputs pair $(\mathsf{Ct}, \mathsf{tag}_{\mathsf{c}})$.

- Sim_1 answers the adaptive query for machine M_j for $j = 1, \ldots, p$, by having on input the master secret key Msk and $z_j = M_j(m)$, in the following way.

  Sim_1 takes secret key sk stored in the state and encrypts the value $c \leftarrow \mathsf{SE.Enc}(sk, z_j)$ and chooses random id_k.

  Sim_1 outputs the pair $(\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}[M_j, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag}_{\mathsf{k}_j}}), \mathsf{tag}_{\mathsf{k}j})$, where $\mathsf{tag}_{\mathsf{k}j} = (id_k, c)$.

We now prove that Sim is a good simulator, which means that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$ and $\mathsf{IdealExp}^{\mathsf{SimFE},\mathcal{A}}_{\mathsf{Sim}}$ are computationally indistinguishable. Recall that, in the random oracle model, these views include all queries made to the random oracle and the responses.

This is proved via a sequence of hybrid experiments as follows.

- Hybrid $H_0^{\mathcal{A}}$: This is the real experiment $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$.
- Hybrid $H_1^{\mathcal{A}}$: This is the real experiment $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$, except that at the beginning of the experiment an additional random secret key $sk' \xleftarrow{R} \{0,1\}^\lambda$ is chosen and the output of random oracle at point $\mathsf{tag}_{\mathsf{c}}$ is programmed as follows. Let $\mathsf{tag}_{\mathsf{k}1} = (id_{k1}, c_1), \ldots, \mathsf{tag}_{\mathsf{k}q} = (id_{kq}, c_q)$ be tags of the tokens returned to adversary at pre-challenge stage. Random oracle is programmed at point $\mathsf{tag}_{\mathsf{c}}$ as follows:

  $$\mathcal{RO}(\mathsf{tag}_{\mathsf{c}}) = \mathsf{SE.Enc}(sk', (id_{k1}, M_1(m)))|| \ldots ||\mathsf{SE.Enc}(sk', (id_{kq}, M_q(m)))$$

  Additionally, the $\mathcal{A}_1$'s key queries for any machine M are answered with tokens, containing tags $\mathsf{tag}_{\mathsf{k}}$ computed as $\mathsf{tag}_{\mathsf{k}} = (id_k, \mathsf{SE.Enc}(sk', M(m)))$. Notice that instead in the previous hybrid experiment $\mathsf{tag}_{\mathsf{k}}$ is computed as $\mathsf{tag}_{\mathsf{k}} = (id_k, c)$ for random id_k and c.

- Hybrid $H_2^{\mathcal{A}}$: We change the way the secret key sk' is chosen. Namely, instead of setting it at random we set $sk' = sk$, where sk is the same secret key encrypted in the third slot of the $\mathsf{MI\text{-}FE}$ ciphertext. In this experiment random oracle is programmed at point $\mathsf{tag}_{\mathsf{c}}$ as follows:

  $$\mathcal{RO}(\mathsf{tag}_{\mathsf{c}}) = \mathsf{SE.Enc}(sk, (id_{k1}, M_1(m)))|| \ldots ||\mathsf{SE.Enc}(sk, (id_{kq}, M_q(m)))$$

- Hybrid $H_3^{\mathcal{A}}$: In this experiment, we change the response to the challenge message m. Instead of taking $m' = (m, 0, sk, \mathsf{Hash}(k_{\mathsf{H}}, \mathcal{RO}(\mathsf{tag}_{\mathsf{c}})), k_{\mathsf{H}})$, we set $m' = (0^{|m|}, 1, sk, \mathsf{Hash}(k_{\mathsf{H}}, \mathcal{RO}(\mathsf{tag}_{\mathsf{c}})), k_{\mathsf{H}})$. Then we compute $c \leftarrow \mathsf{MI\text{-}FE.Enc}(\mathsf{Ek}_1, m')$ and return a pair $(c, \mathsf{tag}_{\mathsf{c}})$. This is the ideal experiment $\mathsf{IdealExp}^{\mathsf{SimFE},\mathcal{A}}_{\mathsf{Sim}}$.

We now show that the relevant distinguishing probabilities between adjacent hybrids are negligible, which completes the proof.

Hybrid $H_0^{\mathcal{A}}$ to Hybrid $H_1^{\mathcal{A}}$: This transition reduces to the ciphertext pseudo-randomness of the underlying SE scheme (See Appendix A.2). For sake of contradiction, suppose there exists a distinguisher $\mathcal{D}$ that distinguishes with non-negligible probability the output distribution of $H_0^{\mathcal{A}}$ and $H_1^{\mathcal{A}}$. Then, $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful adversary $\mathcal{B}$ that breaks the ciphertext pseudo-randomness of SE playing against its oracle (that is either $\mathsf{Encrypt}$ or $\mathcal{O}$, see Appendix A.2). Specifically, $\mathcal{B}$ does the following.

- $\mathcal{B}$ runs $\mathsf{SimFE.Setup}$ honestly obtaining a pair $(\mathsf{Pk}, \mathsf{Msk})$ and runs $\mathcal{A}_0(\mathsf{Pk})$.

  $\mathcal{A}_0$'s key queries for machines $M_1, \ldots, M_q$ are answered honestly by running $\mathsf{SimFE.KeyGen}(\mathsf{Msk}, M_i)$. Let the tags attached to the returned tokens be $\mathsf{tag}_{\mathsf{k}1} = (id_{k1}, c_1), \ldots, \mathsf{tag}_{\mathsf{k}q} = (id_{kq}, c_q)$.

  $\mathcal{B}$ simulates answers to $\mathcal{A}_0$'s random oracle queries. For any input query it responses with random string and stores the output value for following queries.

  Eventually, $\mathcal{A}_0$ outputs challenge message m and the state $\mathtt{st}$.

  $\mathcal{B}$ asks its oracle for encryption of pairs $(id_{k1}, M_1(m)), \ldots, (id_{kq}, M_q(m))$ getting responses $c_1, \ldots, c_q$. Then $\mathcal{B}$ chooses honestly random $\mathsf{tag}_{\mathsf{c}}$ for encryption and programs the simulated $\mathcal{RO}$ by setting $\mathcal{RO}(\mathsf{tag}_{\mathsf{c}}) = c_1 || \ldots || c_q$. $\mathcal{B}$ computes the challenge ciphertext Ct honestly, except that the responded ciphertext tag is $\mathsf{tag}_{\mathsf{c}}$.

- $\mathcal{B}$ runs $\mathcal{A}_1$ with Ct, Pk and state $\mathtt{st}$. For any $\mathcal{A}_1$'s key query for machine M, $\mathcal{B}$ asks its oracle encryption of $M(m)$ getting response c. Then it chooses random id_k and responses $\mathcal{A}_1$'s query with $\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag}_{\mathsf{k}}})$, where $\mathsf{tag}_{\mathsf{k}} = (id_k, c)$. $\mathcal{A}_1$ eventually outputs α.

- $\mathcal{B}$ runs distinguisher $\mathcal{D}$ on input (Pk, m, α) and outputs his guess as its own.

$\mathcal{A}$ asks $\mathcal{RO}$ for $\mathsf{tag}_{\mathsf{c}}$ before the challenge with negligible probability. Now notice that if this event does not happen, then if the oracle of $\mathcal{B}$ returns truly random strings in response to $\mathcal{B}$'s queries then the view of $\mathcal{A}$ is exactly the view in $H_0^{\mathcal{A}}$ and if the oracle returns encryptions of queried values then the view of $\mathcal{A}$ is exactly the view in $H_1^{\mathcal{A}}$. Hence $\mathcal{B}$ breaks the ciphertext pseudo-randomness of SE with non-negligible probability.

Hybrid $H_1^{\mathcal{A}}$ to Hybrid $H_2^{\mathcal{A}}$: The indistinguishability of these two hybrid experiments reduces to CRIND-Security of the starting $\mathsf{MI\text{-}FE}$ scheme with respect to some checker $\mathsf{Checker}$ to be defined later. For sake of contradiction, suppose there exists a distinguisher $\mathcal{D}$ that distinguishes with non-negligible probability the output distribution of $H_1^{\mathcal{A}}$ and $H_2^{\mathcal{A}}$. Then, $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful CRIND adversary $\mathcal{B}$ for $\mathsf{MI\text{-}FE}$ with respect to some checker $\mathsf{Checker}$ that we will define later. Specifically, $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ does the following.

- $\mathcal{B}_0$ on input the public parameters Mpk generated by $\mathsf{MI\text{-}FE.Setup}$, and random r runs $\mathcal{A}_0$ on input public key $\mathsf{Pk} = (\mathsf{Mpk}, r)$.

  Then, $\mathcal{B}_0$ answers any $\mathcal{A}_0$'s token query M_i by using its oracle $\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \cdot)$ as follows: $\mathcal{B}_0$ chooses a random $\mathsf{tag}_{\mathsf{k}i} = (id_{ki}, c_i)$ and outputs the pair $(\mathtt{Tok}_i, \mathsf{tag}_{\mathsf{k}i})$ where $\mathtt{Tok}_i = \mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}[M_i, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag}_{\mathsf{k}i}})$.

  Eventually, $\mathcal{A}_0$ outputs message m and the state $\mathtt{st}$, which are stored in $\mathcal{B}_0$ state.

Then $\mathcal{B}_0$ chooses random sk and sets $k_{\mathsf{H}} = r$, encrypts $C_i \leftarrow \mathsf{SE.Enc}(sk, (id_{ki}, M_i(m)))$ and programs the random oracle by setting $\mathcal{RO}(\mathsf{tag_c}) = C_1 || \ldots || C_q$.

It also chooses random sk' and outputs two challenge messages: $(m, 0, sk, h)$ and $(m, 0, sk', h)$ where $h = \mathsf{Hash}(k_{\mathsf{H}}, \mathcal{RO}(\mathsf{tag_c}))$. $\mathcal{B}_1$ receives as input encryption of one of challenge messages concatenated with $k_{\mathsf{H}} = r$.

- $\mathcal{B}_1$ on input key Mpk, a ciphertext Ct_1 and the state $\mathtt{st}$, runs $\mathcal{A}_1$ on the input $(\mathsf{Pk} = (\mathsf{Mpk}, r), \mathsf{Ct}_1, \mathtt{st})$.

  $\mathcal{B}_1$ answers A_1's adaptive queries for machine M by using its oracle $\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \cdot)$ as follows: chooses a random id_k, computes $c \leftarrow \mathsf{SE.Enc}(sk, M(m))$, sets $\mathsf{tag_k} = (id_k, c)$ and outputs the pair $(\mathtt{Tok}, \mathsf{tag_k})$ where $\mathtt{Tok} = \mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag_k}})$.

  Eventually, $\mathcal{A}_1$ outputs α, then $\mathcal{B}_1$ invokes $\mathcal{D}$ on input $(\mathsf{Pk} = (\mathsf{Mpk}, r), m, \alpha)$ and returns $\mathcal{D}$'s guess as its own.

Notice that if the challenger returns encryption of $(m, 0, sk', h, k_{\mathsf{H}})$ then the view of $\mathcal{A}$ is the same as in $H_1^{\mathcal{A}}$, and if the challenger returns encryption of $(m, 0, sk, h, k_{\mathsf{H}})$ then the view of $\mathcal{A}$ is the same as in $H_2^{\mathcal{A}}$. Hence, if $\mathcal{D}$ distinguishes these two cases then $\mathcal{B}$ guesses the right bit in the game of the CRIND-security of MI-FE, since $sk = sk'$ only with negligible probability. It remains to prove the following claim.

Claim 3.8 There exists an efficient checker $\mathsf{Checker}$ that satisfies the property required in the definition of CRIND-Security and $\mathcal{B}$ is valid with respect to $\mathsf{Checker}$ (cf. remark 2.6).

Notice that the trapdoor machines have a specified format that includes a sub-routine for a CRHF Hash. The checker must only check that this subroutine equals the code of Hash and that the trapdoor machine and the messages have the right format, specifically that the machine has the format of our trapdoor machines with the right sub-routine Hash and that both messages has flag set to normal mode and contain the same hash value h. Note that if this format is guaranteed, then the evaluations of $\mathsf{Trap}[M_i, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag}_{ki}}$ are equal for any second argument. For machines and messages that satisfy these checks, the condition of collision-resistance compatibility is guaranteed and the checker will output true, otherwise it will output false. As a consequence, if the the inputs m_0, m_1 and M to the checker do not satisfy the property of collision-resistance compatibility, it follows that the machine M or the messages can not have the required format, and thus the checker will output false, as it was to prove. Furthermore, by construction, the trapdoor machines queried by $\mathcal{B}$ make the $\mathsf{Checker}$ always output true.

Hybrid $H_2^{\mathcal{A}}$ to Hybrid $H_3^{\mathcal{A}}$: The indistinguishability of these two hybrids reduces to the CRIND-Security (with respect to some checker $\mathsf{Checker}$ with efficient checkability) of the underlying MI-FE scheme. For sake of contradiction, suppose there exists an adversary $\mathcal{D}$ that distinguishes with non-negligible probability the output distributions of $H_2^{\mathcal{A}}$ and $H_3^{\mathcal{A}}$. Then, $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful adversary $\mathcal{B}$ against the CRIND-Security of MI-FE with respect to a checker $\mathsf{Checker}$ that we will define later. Specifically, $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ does the following.

- $\mathcal{B}_0$ on input the public parameters Mpk generated by $\mathsf{MI\text{-}FE.Setup}$ and random r, runs $\mathcal{A}_0$ on input (Mpk, r).

  Then, $\mathcal{B}_0$ answers any $\mathcal{A}_0$'s token query M_i by using its oracle $\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \cdot)$ as follows: $\mathcal{B}_0$ chooses a random $\mathsf{tag}_{ki} = (id_{ki}, c_i)$ and outputs the pair $(\mathtt{Tok}_i, \mathsf{tag}_{ki})$ where $\mathtt{Tok}_i = \mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}[M_i, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag}_{ki}})$.

Eventually, $\mathcal{A}_0$ outputs message m and the state $\mathtt{st}$, which are stored in $\mathcal{B}_0$ state.

Then $\mathcal{B}_0$ sets $k_{\mathsf{H}} = r$, chooses random sk, encrypts $C_i \leftarrow \mathsf{SE.Enc}(sk, (id_{ki}, M_i(m)))$ and programs the random oracle as

$$\mathcal{RO}(\mathsf{tag_c}) = C_1 || \ldots || C_q$$

$\mathcal{B}_0$ outputs a pair of challenge messages: $m'_0 = (m, 0, sk, h)$ and $m'_1 = (0^{|m|}, 1, sk, h)$ where $h = \mathsf{Hash}(k_{\mathsf{H}}, \mathcal{RO}(\mathsf{tag_c}))$. Note that $\mathcal{B}_1$ receives as input encryption of one of challenge messages concatenated with $k_{\mathsf{H}} = r$, i.e., receives an encryption of $m_b = (m'_b || k_{\mathsf{H}})$ for randomly chosen $b \in \{0, 1\}$.

- $\mathcal{B}_1$ on input key Mpk, a ciphertext Ct_1 and the state $\mathtt{st}$, runs $\mathcal{A}_1$ on the input $(\mathsf{Pk} = (\mathsf{Mpk}, r), \mathsf{Ct}_1, \mathtt{st})$.

  $\mathcal{B}_1$ answers A_1's adaptive queries for machine M by using its oracle $\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \cdot)$ as follows: chooses a random id_k, computes $c \leftarrow \mathsf{SE.Enc}(sk, M(m))$, sets $\mathsf{tag_k} = (id_k, c)$ and outputs the pair $(\mathtt{Tok}, \mathsf{tag_k})$ where $\mathtt{Tok} = \mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}[M, \mathsf{Hash}, \mathsf{SE}]^{\mathsf{tag_k}})$.

  Eventually, $\mathcal{A}_1$ outputs α, then $\mathcal{B}_1$ invokes $\mathcal{D}$ on input $(\mathsf{Pk} = (\mathsf{Mpk}, r), m, \alpha)$ and returns $\mathcal{D}$'s guess as its own.

If the challenger returns an encryption of $m_0 = (m'_0 || k_{\mathsf{H}})$ then the view of $\mathcal{A}$ is the same as in $H_2^{\mathcal{A}}$, else if the challenger returns an encryption of $m_1 = (m'_1 || k_{\mathsf{H}})$ then the view of $\mathcal{A}$ is the same as in $H_3^{\mathcal{A}}$. Hence, if $\mathcal{D}$ distinguishes these two cases then $\mathcal{B}$ makes the right guess in the game of the CRIND-Security of $\mathsf{MI\text{-}FE}$. Moreover, the challenges satisfy the requirement of collision-resistance compatibility in the definition CRIND-Security definition because for any queried trapdoor machine Trap, finding second argument R such that

$$\mathsf{Trap}(m_0 = (m'_0 || k_{\mathsf{H}}), R) \neq \mathsf{Trap}(m'_1 = (m_1 || k_{\mathsf{H}}), R)$$

implies finding $R \neq \mathcal{RO}(\mathsf{tag_c})$ such that $\mathsf{Hash}(k_{\mathsf{H}}, R) = \mathsf{Hash}(k_{\mathsf{H}}, \mathcal{RO}(\mathsf{tag_c}))$. It is easy to see that this contradicts the security of the collision-resistance hash function Hash, because k_{H} is chosen at random, and therefore will occur with negligible probability. This is proven in the following claim.

It remains to prove the following claim.

Claim 3.9 Let K denote the entire set of key queries made by adversary $\mathcal{B}$. Then, w.v.h.p., the challenge messages m'_0 and m'_1 chosen by $\mathcal{B}$ must be collision-resistant compatible with K.

Suppose toward a contradiction that this does not hold. Then there exists an adversary E that breaks the security of Hash. Such adversary E receives k_{H} and generates the trapdoor machine Trap along with m_0 and m_1 as described before, in particular setting the value h in both messages to $\mathsf{Hash}(k_{\mathsf{H}}, \mathcal{RO}(\mathsf{tag_c}))$. Suppose that E outputs a messages R such that $\mathsf{Trap}(m_0, R) \neq \mathsf{Trap}(m_1, R)$. By definition of Trap, m_0 and m_1, it holds that either $\mathsf{Trap}(m_0, R) = \mathsf{Trap}(m_1, R) = \perp$ or $\mathsf{Trap}(m_0, R) \neq \perp$ and $\mathsf{Trap}(m_1, R) \neq \perp$. By definition of Trap, this implies that R is such that $\mathsf{Hash}(k_{\mathsf{H}}, R) = h$. Moreover, by construction it holds that $\mathsf{Trap}(m_0, \mathcal{RO}(\mathsf{tag_c})) = \mathsf{Trap}(m_1, \mathcal{RO}(\mathsf{tag_c}))$. Therefore, $R \neq \mathcal{RO}(\mathsf{tag_c})$ and $\mathsf{Hash}(k_{\mathsf{H}}, R) = \mathsf{Hash}(k_{\mathsf{H}}, \mathcal{RO}(\mathsf{tag_c}))$, contradicting the security of Hash.

Claim 3.10 There exists an efficient checker $\mathsf{Checker}$ that satisfies the property required in the definition of CRIND-Security and $\mathcal{B}$ is valid with respect to $\mathsf{Checker}$ (cf. remark 2.6).

Notice that the trapdoor machines have a specified format that includes a sub-routine for a CRHF Hash. The checker must only check that this subroutine equals the code of Hash and that the trapdoor machine and the messages have the right format, specifically that the machine has the format of our trapdoor machines with the right sub-routine Hash and that one of the messages has flag set to normal mode and the other one has flag set to trapdoor mode and both messages have the same hash value h and same secret-key sk. For machines and messages that satisfy these checks, the condition of collision-resistance compatibility is guaranteed and the checker will output true, otherwise it will output false. As a consequence, if the the inputs m_0, m_1 and M to the checker do not satisfy the property of collision-resistance compatibility, it follows that the machine M or the messages can not have the required format, and thus the checker will output false, as it was to prove. Furthermore, by construction, the trapdoor machines queried by $\mathcal{B}$ make the Checker always output true.

∎

Remark 3.11 The reader may have noticed that the security of the second ciphertext guaranteed by the underlying MI-FE is not *necessary*. That is, our transformation would work even assuming 2-inputs MI-FE systems that take the second input *in clear*. Moreover, the theorem would hold even assuming a MI-FE scheme CRIND-Secure only with respect to adversaries asking bounded non-adaptive and ciphertext queries and unbounded adaptive queries, and satisfying the same efficiency requirements. This holds for the transformation of Section 3.2 as well.

3.2 (q_1, q_c, q_2)-SIM-Security with short tokens

The previous transformation suffers from the problem that the size of the tokens grows as the number of ciphertext queries q_c. In this Section we show how to achieve (q_1, q_c, q_2)-SIM-Security. Notice that in the standard model constructions satisfying this level of security like the scheme of Gorbunov *et al.* [GVW12b] have short ciphertexts and tokens. Moreover, De Caro and Iovino [CI13] showed an impossibility result for this setting. Our transformation assumes a 3-inputs MI-FE scheme (CRIND-Secure in the standard model). The resulting scheme is (q_1, q_c, q_2)-SIM-Secure according to the definition with simulated setup (cf. Remark C.6). The idea is very similar to the transformation presented in Section 3.1.

Sketch of the transformation. The tokens will be symmetrical to the the ciphertexts. Specifically, a token in our scheme consists of a token T for a 3-inputs MI-FE scheme and a tag $\mathsf{tag_k}$ with the following changes. The token T has an additional slot containing the hash of $\mathcal{RO}(\mathsf{tag_k})$ and an identifier id_t and the ciphertext has a additional slot containing an identifier id_c. In normal mode both identifiers are set to random strings. In the simulated experiment, the identifiers will be ciphertexts encrypted with the same secret-key used in the other slot of the ciphertext. Let us denote by id'_t and id'_c the values encrypted in id_t and id_c. In trapdoor mode, the functionality checks whether $id'_c > id'_t$. In such case the functionality works as before taking the output from the RO programmed at point $\mathcal{RO}(\mathsf{tag_k})$. Instead, if $id'_c < id'_t$, the functionality takes the output from $\mathcal{RO}(\mathsf{tag_c})$. In both normal and trapdoor mode, the functionality checks whether both hash values are correct (that is, it checks whether the hash value in the token equals the hash of $\mathcal{RO}(\mathsf{tag_k})$ and that the hash value in the ciphertext equals the hash of $\mathcal{RO}(\mathsf{tag_c})$) and if one of the tests does not pass, the functionality returns error. During the simulation, these identifiers will contain a temporal index increasing over time (that is the i-th token or ciphertext will contain an identifier that encrypts i). Notice that since we need to simulate tokens (to set the identifiers to the encryption of the correct timestamp), we need to weaken the security to allow simulated setup (cf. Remark C.6).

The token T of our MI-FE scheme is for a machine Trap that takes 3 inputs: the message (extended as usual with the additional slots), and two random strings R_c and R_k, which are used to decrypt in trapdoor mode, and works in the obvious way by executing Trap on the three inputs. Finally, in the simulated experiment, the RO is programmed to output a fixed number $\max q_1, q_c, q_2$ of ciphertexts, value that is proportional to the running-times of the procedures as well.

Security. The security proof is identical to that of our main transformation with few changes. We first need to switch the slot of the identifiers in both ciphertexts and tokens to be encryption of the the temporal index. Then we proceed as before.

3.3 $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-SIM-Security in the Timestamp model

We recall that any known impossibility results in the standard model also apply to the symmetric-key setting. In this Section we show how to achieve unbounded SIM-Security in the RO model in a variant of the symmetric-key setting that we call the *timestamp* model. Moreover, our scheme enjoys ciphertexts and tokens of constant size. The timestamp model is identical to the symmetric-key mode except for the following changes:

- The encryption and key generation procedures also take as input a *temporal index* or *timestamp*. The security of this index is not required. The security experiments are identical to those of the symmetric-key model except that the queries are answered by providing tokens and ciphertexts with *increasing* temporal index (the exact value does not matter until as long as are ordered in order of invocation). Roughly speaking, this is equivalent to saying that the procedures are stateful. Notice that in the symmetric-key model, this change has no cost since the user who set-up the system can keep the value of the current timestamp and guarantee that ciphertexts and tokens are generated with timestamps of increasing order.

- For simplicity, we also assume that there is a decryption key. Precisely, the evaluation algorithm takes as input a token, a ciphertext and a decryption key. It is easy to see that this decryption key can be removed at the cost of including it in any token or ciphertext.

Sketch of the transformation. With these changes in mind it is easy to modify the scheme of Section 3.2 to satisfy $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-$\mathsf{SIM}$-Security in the RO model. Precisely, the slots for the identifiers will contain the temporal index in *clear*[12]. In the scheme, the tags will be such that the (both non-programmed and programmed) RO on these input will output a string of size proportional to i, where i is the temporal index. This can be done by assuming that the RO outputs a single bit and invoking it many times. That is, instead of computing $\mathcal{RO}(\mathsf{tag_c})$, the procedures will compute $\mathcal{RO}(\mathsf{tag_c}||j)$ for $j = 1, \ldots, m$ where m is the needed size. For simplicity, henceforth, we assume that the RO outputs strings of variable-length. As byproduct, we need to program the RO on the tag $\mathsf{tag_c}$ (resp. $\mathsf{tag_k}$) of a ciphertext (resp. token) with timestamp i to only output i ciphertexts. Thus, we do not need to fix in *advance* any bound, which was the only limitation of the previous transformations. Notice that the evaluation algorithm needs to encrypt the output of $\mathcal{RO}(\mathsf{tag_c})$ and $\mathcal{RO}(\mathsf{tag_k})$ and this is done by using the encryption keys Ek_2 and Ek_3 for the second and third input. This is the reason why we assume that there is a decryption key that in this scheme consists of the pair $(\mathsf{Ek}_2, \mathsf{Ek}_3)$.

Security. The security proof is identical to that of the transformation of Section 3.2 with further simplifications due to the fact that we do not need to have the temporal index encrypted.

[12] We stress that we could also assume that the temporal index is appended in clear to the final ciphertext.

We first need to switch the slot of the identifiers in both ciphertext and tokens to be encryption of the the temporal index. Then the proof proceeds as before.

4 Constructions of CRIND-Secure MI-FE from e$\mathcal{O}$

Overview. In order to achieve CRIND-security of MI-FE (as defined in Section 2.3), we make use of the following ideas inspired by the construction of fully IND-Secure FE of Boyle *et al.* [BCP14]. We assume a functional signature scheme FS [BGI13]. Namely, our encryption procedure takes as input the first input m_1 and produces an obfuscation of a machine that has embedded m_1 and takes as input a second message m_2 and a functional signature for some function f and (1) verifies the signature and (2) outputs $f(m_1, m_2)$. Roughly speaking, we want prevent the adversary to be able to find distinguishing inputs. To this scope, we need to forbid the adversary from evaluating the machine on functions for which it did not see a signature. For the same reasons as in Boyle *et al.* it is not possible to use a standard signature scheme. This is because, the adversary $\mathcal{A}$ against e$\mathcal{O}$ needs to produce a view to the adversary $\mathcal{B}$ against CRIND-Security, and in particular to simulate the *post-challenge* tokens. To that aim, $\mathcal{A}$ would need to receive an auxiliary input z containing the signing key of the traditional signature scheme but in this case an extractor with access to z could easily find a distinguishing input. As in Boyle *et al.* we resort to functional signatures. We recall their ideas. In the scheme, they put a functional signing key that allows to sign any function. In the security proof, they use the property of function privacy to show that the original experiment is computationally indistinguishable to an experiment where the post-challenge queries are answered with respect to a *restricted* signing key for the Boolean predicate that is verified on all and only the machines T for which $T(m_0) = T(m_1)$, where m_0 and m_1 are the challenges chosen by the adversary against IND-Security. Thus, putting this restricted signing key in the auxiliary distribution does *not* hurt of the security since an extractor can not make use of it to find a distinguishing input. In the case of CRIND-Security, it is not longer true that $T(m_0) = T(m_1)$ but we will invoke the properties of the checker and we set the Boolean predicate to one that is verified for all machines T approved by the checker with respect to the challenges, i.e., such that $\mathsf{Checker}(T, m_0, m_1) = 1$. Then, by the property of the checker and valid adversaries, it follows that for any machine T for which the valid adversary asked a query, it is difficult to find a second input m_2 such that $T(m_0, m_2) \neq T(m_1, m_2)$. Thus, the existence of an extractor for our distribution would contradict the hypothesis that the adversary is valid and only asked queries for machines and challenges satisfying the collision-resistance compatibility. Precisely, we have the following transformation.

Definition 4.1 [e$\mathcal{O}$-Based Transformation]
Let e$\mathcal{O}$ be an extractability obfuscator w.r.t. distributional auxiliary input (cf. Remark 2.9). Let $\mathsf{FS} = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Verify})$ be a signature scheme.

For any message m_1 and verification key vk of FS, let us define a machine M_{m_1} (where for simplicity we omit the other parameters in subscript) that takes two inputs, a signature σ_T of machine T and a message m_2, and performs the following:

- The machine verifies the signature σ_T according to vk, and if it is an invalid signature, it returns $\perp$;
- The machine returns $T(m_1, m_2)$.

We define a new 2-inputs functional encryption scheme

$\mathsf{CRFE}[\mathsf{e}\mathcal{O}, \mathsf{FS}] = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality TM_2 as follows[13]

- $\mathsf{Setup}(1^\lambda)$: chooses a pair $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$ and generates a key $\mathsf{sk}_1 \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, 1)$ that allows signing all messages (i.e., for the always-accepting predicate $1(T) = T \ \forall \ T$). It sets $\mathsf{Ek}_1 = \mathsf{Ek}_2 = \mathsf{vk}$ and outputs $\mathsf{Mpk} = \mathsf{Ek}_1$ and $\mathsf{Msk} = (\mathsf{sk}_1, \mathsf{vk})$.

- $\mathsf{Enc}(\mathsf{Ek}, m)$: depending on whether Ek is an encryption key for first or second input:

  - if $\mathsf{Ek} = \mathsf{Ek}_1$ then outputs $\mathsf{e}\mathcal{O}(M_m)$ where M_m is defined as above with respect to m and vk (recall that for simiplicity we omit the subscript for vk).
  - if $\mathsf{Ek} = \mathsf{Ek}_2$ then outputs the message m in clear (recall that we are not interested in the security of the second input and we adopted the formalism of multi-input FE to avoid the need of a new syntax and for sake of generality, e.g., providing in future constructions that satisfy *both* IND-Security and CRIND-Security).

- $\mathsf{KeyGen}(\mathsf{Msk}, T)$: on input $\mathsf{Msk} = (\mathsf{sk}_1, \mathsf{vk})$ and a machine T, the algorithm generates a signature on T via $\sigma_T \leftarrow \mathsf{Signature.Sign}(\mathsf{sk}_1, \mathsf{T})$ and outputs token σ_T.

- $\mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}_1, \mathsf{Ct}_2, \mathtt{Tok})$: on input $\mathsf{Mpk} = \mathsf{vk}$, Ct_1 which is an obfuscated machine M of machine M_{m_1}, $\mathsf{Ct}_2 = m_2$, $\mathtt{Tok} = \sigma_T$, runs machine M with the other inputs as arguments, and returns the machine output as its own.

Correctness. It is easy to see that the scheme satisfies correctness assuming the correctness of $\mathsf{e}\mathcal{O}$ and FS.

We now analyze the security of the constructed scheme. The reader is encouraged to compare this with the proof of fully IND-Secure (single-input) FE of Boyle *et al.* that is very similar except for few changes.

Theorem 4.2 *If $\mathsf{e}\mathcal{O}$ is an extractable obfuscator w.r.t. distributional auxiliary input,* FS *is an unforgeable functional signature scheme with function privacy, then, for any* Checker *satisfying the requirement of the* CRIND*-Security, it holds that* $\mathsf{CRFE}[\mathsf{e}\mathcal{O}, \mathsf{FS}]$ *is* CRIND*-Secure. Furthermore such scheme satisfies input-specific run time and assuming that* FS *is also succinct, so* CRFE *does.*

Proof: We define the following hybrids. Let $q(\lambda)$ be a bound on the number of *post-challenge* token queries asked by $\mathcal{B}$ in any execution with security parameter 1^λ. Such bound exists because $\mathcal{B}$ is a PPT algorithm.

- Hybrid $H_0^{\mathcal{B}}$: This is the real experiment $\mathsf{CRIND}_{\mathcal{B}}^{\mathsf{CRFE}[\mathsf{e}\mathcal{O},\mathsf{FS}]}$.

- Hybrid $H_i^{\mathcal{B}}, i = 0, \ldots, q$: Same as the previous hybrid, except that the first i post-challenge token queries are answered with respect to a *restricted* signing key sk_C for the Boolean predicate C that allows one to sign exactly Turing machines T for which $\mathsf{Checker}(T, m_0, m_1) = 1$. (This is one of the differences with the proof of Boyle *et al.* wherein, being the scope to prove IND-Security, the signing key is for a predicate that allows one to sign exactly the machines T for which $T(m_0) = T(m_1)$. This is not possible in our case, but we make use of the definition of valid adversary that dictates that

[13] For simplicity, henceforth we omit to specify whether the functionality is with respect to machine of fixed time or input-specific. Both cases can be taken in account with small changes.

such adversary will only make queries for machines approved by the checker.). Specifically, at the beginning of the game the challenger generates a restricted signing key $\mathsf{sk}_\mathsf{C} \leftarrow \mathsf{FS.KeyGen}(\mathsf{Msk}, \mathsf{C})$. The pre-challenge queries are answered using the standard signing key sk_1 as in hybrid H_0. The first i post-challenge token queries are answered using the restricted key sk_C, that is a token query for machine T is answered with $\sigma_T \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_\mathsf{C}, \mathsf{T})$. All remaining token queries are anwered using the standard key sk_1.

Claim 4.3 For $i = 1, \ldots, q$, the advantage of $\mathcal{B}$ in guessing the bit b in hybrid $H_i^{\mathcal{B}}$ is equal to the advantage of $\mathcal{B}$ in guessing the bit b in hybrid $H_{i-1}^{\mathcal{B}}$ up to a negligible factor.

We prove the claim by using the function privacy property of FS. Namely, for any $i \in [q]$, consider the following adversary $\mathcal{A}_{\mathsf{priv}}(1^\lambda)$ against function privacy of FS.

- $\mathcal{A}^i_{\mathsf{priv}}$ is given keys $(\mathsf{vk}, \mathsf{msk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$ from the function privacy challenger.
- $\mathcal{A}^i_{\mathsf{priv}}$ submits the all-accepting function 1 as the first of its two challenge functions, and receives a corresponding signing key $\mathsf{sk}_1 \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, 1)$.
- $\mathcal{A}^i_{\mathsf{priv}}$ simulates interaction with $\mathcal{B}$. First, it forwards vk to $\mathcal{B}$ as the public-key and chooses a random string $r \in \{0,1\}^\lambda$. For each token query T made by $\mathcal{B}$, it generates a signature on T using key sk_1.
- $\mathcal{A}^i_{\mathsf{priv}}$ At some point $\mathcal{B}$ outputs a pair of messages m_0, m_1. $\mathcal{A}^i_{\mathsf{priv}}$ generates a challenge ciphertext in the CRIND-Security game by sampling a random bit b and encrypting $(m_b||r)$ and sending it to $\mathcal{B}$.
- $\mathcal{A}^i_{\mathsf{priv}}$ submits as its second challenge function C (as defined above). It receives a corresponding signing key $\mathsf{sk}_\mathsf{C} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{P}_\mathsf{C})$.
- $\mathcal{A}^i_{\mathsf{priv}}$ now simulates interaction with $\mathcal{B}$ as follows.

  For the first $i-1$ post-challenge token queries T made by $\mathcal{B}$, $\mathcal{A}^i_{\mathsf{priv}}$ generates a signature using key sk_C, i.e., $\sigma_T \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_\mathsf{C}, \mathsf{T})$. For $\mathcal{B}$'s i-th post-challenge query, $\mathcal{A}^i_{\mathsf{priv}}$ submits the pair of preimages (T, T) to the function privacy challenger (note that $1(T) = C(T) = T$) since, being $\mathcal{B}$ a valid adversary, it only asks queries T such that $\mathsf{Checker}(T, m_0, m_1) = 1$), and receives a signature σ_T generated either using key sk_1 or key sk_C. $\mathcal{A}^i_{\mathsf{priv}}$ generates the remaining post-challenge queries of $\mathcal{B}$ using key sk_1.
- Eventually $\mathcal{B}$ outputs a bit b'. If $b' = b$ is a correct guess, then $\mathcal{A}^i_{\mathsf{priv}}$ outputs function 1; otherwise, it outputs function C.

Note that if the function privacy challenger selected the function 1, then $\mathcal{A}^i_{\mathsf{priv}}$ perfectly simulates hybrid $H^{\mathcal{B}}_{i-1}$, otherwise it perfectly simulates hybrid $H^{\mathcal{B}}_i$. Thus, the advantage of $\mathcal{A}^i_{\mathsf{priv}}$ is exactly the difference in guessing the bit b in the two hybrids, $H^{\mathcal{B}}_i$ and $H^{\mathcal{B}}_{i-1}$ and the claim follows from the function privacy property.

Next, we define the following distribution $\mathcal{D}$ depending on $\mathcal{B}$.

- $\mathcal{D}(1^\lambda)$ gets $r \xleftarrow{R} \{0,1\}^\lambda$, samples a key pair $(\mathsf{vk}, \mathsf{msk} \leftarrow \mathsf{FS.Setup}(1^\lambda)$ and generates the signing key for the all-accepting function 1 by $\mathsf{sk}_1 \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, 1)$.

- Using $\mathsf{sk_1}$ and vk, $\mathcal{D}$ simulates the action of $\mathcal{B}$ in experiment $H_q^{\mathcal{B}}$ up to the point in which $\mathcal{B}$ outputs a pair of challenge messages m_1, m_1. Denote by $\mathsf{view}_{\mathcal{B}}$ the current view of $\mathcal{B}$ up to this point of the simulation.
- $\mathcal{D}$ generates a signing key $\mathsf{sk_C}$ for the function C as defined above and machines $M_{m_0||r}$ and $M_{m_1||r}$ as defined above (recall that as usual we omit to specify the subscript relative to vk).
- $\mathcal{D}$ outputs the tuple $(M_{m_0||r}, M_{m_1||r}, z = (\mathsf{view}_{\mathcal{B}}, \mathsf{sk_C}))$.

We now can construct an adversary $\mathcal{A}(1^\lambda, M', M_0, M_1, z)$ against the security of $\mathsf{e}\mathcal{O}$.

- $\mathcal{A}$ takes as input the security parameter 1^λ, an obfuscation M' of machine M_b for randomly chosen bit b, two machines M_0 and M_1, and auxiliary input $z = (\mathsf{view}_{\mathcal{B}}, \mathsf{sk_C})$.
- Using $\mathsf{view}_{\mathcal{B}}$, $\mathcal{A}$ returns $\mathcal{B}$ to the state of execution as in the corresponding earlier simulation during the $\mathcal{D}$ sampling process.
- Simulate the challenge ciphertex to $\mathcal{B}$ as M'. For each subsequent token query M made by $\mathcal{B}$, $\mathcal{A}$ answers it by producing a signature on M using $\mathsf{sk_C}$.
- Eventually, $\mathcal{B}$ outputs a bit b' for the challenge ciphertext that $\mathcal{A}$ returns as its own guess.

Note that the interaction with the adversary $\mathcal{B}$ in sampling from $\mathcal{D}$ is precisely a simulation in hybrid $H_q^{\mathcal{B}}$ up to the point in which $\mathcal{B}$ outputs the challenge messages,and the interaction with $\mathcal{B}$ made by $\mathcal{A}$ is precisely a simulation of the remaining steps in hybrid $H_q^{\mathcal{B}}$. We are assuming that the advantage of $\mathcal{B}$ in hybrid H_q^i is $\geq 2a(\lambda)$ for some non-negligible function $a(\lambda)$. This implies that there is a polynomial $p(\lambda)$ such that for an infinite set S of values λ, it holds that the advantage of $\mathcal{B}$ in hybrid H_q^i for parameter λ is greater than $1/2p(\lambda)$. Thus, by an averaging argument, for all $\lambda \in S$, $\mathcal{A}$'s advantage (with respect to λ) in guessing the bit b on which it is challenged upon is greater than $1/p$ with probability greater than $1/p$ over the output of $\mathcal{D}$. By the security of $\mathsf{e}\mathcal{O}$ this implies a corresponding PPT extractor E and polynomial $q(\lambda)$ and negligible function $\mathsf{negl}(\lambda)$ such for all $\lambda \in S$, with probability $1 - \mathsf{negl}(\lambda)$ over the output (M_0, M_1, z) of $\mathcal{D}$, it holds that:
if $\Pr\left[b \xleftarrow{R} \{0,1\}; M' \leftarrow \mathsf{e}\mathcal{O}(1^\lambda, M_b) : \mathcal{A}(1^\lambda, M', M_0, M_1, z) = b \right] \geq \frac{1}{2} + \frac{1}{p(\lambda)}$,
then $\Pr\left[w \leftarrow \mathsf{E}(1^\lambda, M_0, M_1, z) : M_0(w) \neq M_1(w) \right] \geq 1/q(\lambda)$. This implies that for an infinite number of values λ, with probability $\geq 1/p(\lambda) - \mathsf{negl}(\lambda)$ over the output (M_0, M_1, z) of $\mathcal{D}$, it holds that $\Pr\left[w \leftarrow \mathsf{E}(1^\lambda, M_0, M_1, z) : M_0(w) \neq M_1(w) \right] \geq 1/q(\lambda)$.

We now show that such PPT extractor can not exist.

Claim 4.4 There can not exist a PPT extractor as above.

Suppose toward a contradiction that there exists such extractor that outputs a signature σ_A for some machine A , and a second input m_2 that distinguishes $M_{m_0||r}$ from $M_{m_1||r}$. We note that any signature output by the extractor must be a valid signature for a machine A for which the adversary asked a query. This follows from the unforgeability of FS. From this fact, and from the fact that the checker approved the triple (A, m_0, m_1), it follows that m_0 and m_1 are collision-resistant compatible with $\{A\}$. Therefore, this adversary can be used to break

the collision-resistance compatibility with respect to m_0 and m_1 and $\{A\}$, contradicting the hypothesis.

It is trivial to see that the claim on input-specific run time holds if the the scheme is used with Turing machines of input-specific run time and that the claim on the succinctness follows easily from our construction and the succinctness of FS. This concludes the proof. ∎

5 Acknowledgments

We thank Abhishek Jain, Adam O'Neill, Anna Sorrentino and the anonymous reviewers for useful comments. Part of this work was done while Vincenzo Iovino was at the University of Warsaw. This work was supported by the WELCOME/2010-4/2 grant founded within the framework of the EU Innovative Economy Operational Programme.

References

[AAB+13] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. Function private functional encryption and property preserving encryption : New definitions and positive results. Cryptology ePrint Archive, Report 2013/744, 2013. http://eprint.iacr.org/.

[AGK+13] Daniel Apon, Dov Gordon, Jonathan Katz, Feng-Hao Liu, Hong-Sheng Zhou, and Elaine Shi. Personal Communication, July 2013.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO (2)*, pages 500–518, 2013.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73. Springer, 2014.

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.

[BO13] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *Cryptology and Network Security - 12th International Conference, CANS 2013, Paraty, Brazil, November 20-22. 2013. Proceedings*, pages 218–234, 2013.

[BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. http://eprint.iacr.org/.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[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.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Berlin, Germany.

[CG13] 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*. Springer, 2013.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press.

[CI13] Angelo De Caro and Vincenzo Iovino. On the power of rewinding simulators in functional encryption. *IACR Cryptology ePrint Archive*, 2013:752, 2013.

[DIJ+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 Canetti and Garay [CG13], pages 519–535.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 308–317. IEEE Computer Society, 1990.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent 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.

[GGHW13] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Cryptology ePrint Archive, Report 2013/860, 2013. `http://eprint.iacr.org/`.

[GGJS13] Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727, 2013. `http://eprint.iacr.org/`.

[GKL+13] S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. *IACR Cryptology ePrint Archive*, 2013:774, 2013.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Canetti and Garay [CG13], pages 536–553.

[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.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/`.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 591–608, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 218–235, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.

[Wee09] Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 417–434, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.

A Standard Notions

A.1 Collision-resistant Hash Functions

Definition A.1 [Collision-resistant Hash Functions] We say that a pair of PPT algorithms $(\mathsf{Gen}, \mathsf{Hash})$ is *collision-resistant hash function* (CRHF in short) if:

- $\mathsf{Gen}(1^\lambda)$ outputs a key s.
- There exists some polynomial $l(\lambda)$ such that Hash on input 1^λ and $x \in \{0,1\}^\star$ outputs a string $y \in \{0,1\}^{l(\lambda)}$. If $H(s,\cdot)$ is only defined for inputs x of length $l'(\lambda)$, where $l'(\lambda) > l(\lambda)$, we say that $(\mathsf{Gen}, \mathsf{Hash})$ is a fixed-length collision-resistant hash function for inputs of length l'.
- It holds that for any (possibly, non-uniform) PPT adversary $\mathcal{A}$ the probability of winning in the following game is negligible in λ:

  1. $s \leftarrow \mathsf{Gen}(1^\lambda)$;
  2. $(x_0, x_1) \leftarrow \mathcal{A}(1^\lambda, s)$;
  3. **Output:** $\mathcal{A}$ wins iff $\mathsf{Hash}(s, x_0) = \mathsf{Hash}(s, x_1)$ and $x_0 \neq x_1$.

A.2 Symmetric-key encryption

Definition A.2 [Symmetric-Key Encryption Scheme] A symmetric-key encryption scheme with key-space $\{0,1\}^s$, message-space $\{0,1\}^\mu$, and ciphertext-space $\{0,1\}^\nu$ is a pair of probabilistic algorithms $(\mathsf{Enc}, \mathsf{Dec})$, such that $\mathsf{Enc} : \{0,1\}^s \times \{0,1\}^\mu \to \{0,1\}^\nu$, $\mathsf{Dec} : \{0,1\}^s \times \{0,1\}^\nu \to \{0,1\}^\mu$, and for every key $K \in \{0,1\}^s$ and every message M,

$$\Pr\left[\, \mathsf{Dec}(K, \mathsf{Enc}(K, M)) = M \,\right] = 1$$

where the probability is taken over the randomness of the algorithms.

Definition A.3 [Ciphertext Pseudo-randomness] A symmetric-key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ has *pseudo-random ciphertexts* if for every PPT adversary $\mathcal{A}$ we have that the following quantity is negligible in s:

$$\left| \Pr\left[\mathcal{A}^{\mathsf{Enc}(K,\cdot)} = 1 \mid K \leftarrow \{0,1\}^s \right] - \Pr\left[\mathcal{A}^{\mathcal{O}(1^s,\cdot)} = 1 \right] \right| \leq \mathsf{negl}(s) \ ,$$

where $O(1^s, \cdot)$ is a randomized oracle that on input message M returns a string independently and uniformly chosen in $\{0,1\}^{\nu(s)}$.

B Functional Signature Schemes

Definition B.1 [Functional Signatures]

A *functional signature scheme* for a message space $\mathcal{M}$, and function family $\mathcal{F} = \{f \colon \mathcal{D}_f \to \mathcal{M}\}$ consists of algorithms $(\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Verify})$:

- $\mathsf{FS.Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{mvk})$: the setup algorithm takes as input the security parameter and outputs the master signing key and master verification key.

- $\mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{f}) \to \mathsf{sk_f}$: the KeyGen algorithm takes as input the master signing key and a function $f \in \mathcal{F}$ (represented as a Boolean circuit), and outputs a signing key for f.
- $\mathsf{FS.Sign}(\mathsf{f}, \mathsf{sk_f}, \mathsf{m}) \to (\mathsf{f}(\mathsf{m}), \sigma)$: the signing algorithm takes as input the signing key for a function $f \in \mathcal{F}$ and an input $m \in \mathcal{D}_f$, and outputs $f(m)$ and a signature of $f(m)$.
- $\mathsf{FS.Verify}(\mathsf{mvk}, \mathsf{m}^*, \sigma) \to \{0, 1\}$: the verification algorithm takes as input the master verification key mvk, a message m and a signature and outputs 1 if the signature is valid.

We say that FS is a functional signature scheme for *unbounded-length* messages if $\mathcal{M} = \{0,1\}^*$. We require the following conditions to hold:

Correctness: For all $f \in \mathcal{F}$ and $m \in \mathcal{D}_f$ holds
$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda), \mathsf{sk_f} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{f}), (\mathsf{m}^*, \sigma) \leftarrow \mathsf{FS.Sign}(\mathsf{f}, \mathsf{sk_f}, \mathsf{m})$
$\implies \mathsf{FS.Verify}(\mathsf{mvk}, \mathsf{m}^*, \sigma) = 1$.

Unforgeability: The scheme is unforgeable if the advantage of any PPT algorithm $\mathcal{A}$ in the following game is negligible:

- The challenger generates $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$, and gives mvk to $\mathcal{A}$
- The adversary is allowed to query a key generation oracle $\mathcal{O}_{key}$, and a signing oracle $\mathcal{O}_{sign}$, that share a dictionary indexed by tuples (f, i), whose entries are signing keys: $\mathsf{sk_f^i} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{f})$. This dictionary keeps track of the keys that have been previously generated during the unforgeability game. The oracles are defined as follows:
  - $\mathcal{O}_{key}(f, i)$:
    - if there exists an entry for the key (f, i) in the dictionary, then output the corresponding $\mathsf{sk_f^i}$.
    - otherwise, sample a fresh key $\mathsf{sk_f^i} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{f})$, add an entry $(f, i) \to \mathsf{sk_f^i}$ to the dictionary, and output sk_f^i.
  - $\mathcal{O}_{sign}(f, i, m)$:
    - if there exists an entry for the key (f, i) in the dictionary, then generate a signature on $f(m)$ using this key: $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{f}, \mathsf{sk_f^i}, \mathsf{m})$.
    - otherwise, sample a fresh key $\mathsf{sk_f^i} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{f})$, add an entry $(f, i) \to \mathsf{sk_f^i}$ to the dictionary, and generate a signature on $f(m)$ using this key: $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{f}, \mathsf{sk_f^i}, \mathsf{m})$.
- The adversary wins if it can produce (m^*, σ) such that
  - $\mathsf{FS.Verify}(\mathsf{mvk}, \mathsf{m}^*, \sigma) = 1$.
  - there does not exist m such that $m^* = f(m)$ for any f which was sent as a query to the $\mathcal{O}_{key}$ oracle.
  - there does not exist a (f, m) pair such that (f, m) was a query to the $\mathcal{O}_{sign}$ oracle and $m^* = f(m)$.

Function privacy: Intuitively, we require the distribution of signatures on a message m' generated via different keys $\mathsf{sk_f}$ to be computationally indistinguishable, *even given the secret keys and master signing key.* Namely, the advantage of any PPT adversary in the following game is negligible:

- The challenger honestly generates a key pair $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$ and gives both values to the adversary. (Note wlog this includes the randomness used in generation).
- The adversary chooses a function f_0 and receives an (honestly generated) secret key $\mathsf{sk}_{\mathsf{f}_0} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{f}_0)$.
- The adversary chooses a second function f_1 for which $|f_0| = |f_1|$ (where padding can be used if there is a known upper bound) and receives an (honestly generated) secret key $\mathsf{sk}_{\mathsf{f}_1} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, \mathsf{f}_1)$.
- The adversary chooses a pair of values (m_0, m_1) for which $|m_0| = |m_1|$ and $f_0(m_0) = f_1(m_1)$.
- The challenger selects a random bit $b \xleftarrow{R} \{0,1\}$ and generates a signature on the image message $m' = f_0(m_0) = f_1(m_1)$ using secret key $\mathsf{sk}_{\mathsf{f}_\mathsf{b}}$, and gives the resulting signature $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_{\mathsf{f}_\mathsf{b}}, \mathsf{m}_\mathsf{b})$ to the adversary.
- The adversary outputs bit b', and wins the game if $b' = b$.

Succinctness: The size of a signature $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_\mathsf{f}, \mathsf{m})$ is bounded by a polynomial in security parameter λ, and the size of the output $|f(m)|$. In particular, it is independent of $|m|$, the size of the input to the function, and $|f|$, the size of a description of the function f.

C FE and its IND-Security

Let us now define the notion of a functional encryption scheme FE for a functionality F.

Definition C.1 [Functional Encryption Scheme] A *functional encryption* scheme FE for functionality F is a tuple $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ of 4 algorithms with the following syntax:

1. $\mathsf{Setup}(1^\lambda)$ outputs *public* and *master secret* keys $(\mathsf{Pk}, \mathsf{Msk})$ for *security parameter* λ.
2. $\mathsf{KeyGen}(\mathsf{Msk}, k)$, on input a master secret key Msk and *key* $k \in K$ outputs *token* $\mathtt{Tok}$.
3. $\mathsf{Enc}(\mathsf{Pk}, m)$, on input public key Pk and *plaintext* $m \in M$ outputs *ciphertext* Ct;
4. $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$ outputs $y \in \Sigma \cup \{\perp\}$.

In addition we make the following *correctness* requirement: for all $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $k \in K$ and $m \in M$, for $\mathtt{Tok} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, m)$, we have that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}) = F(k, m)$ whenever $F(k, m) \neq \perp$, except with negligible probability. (See [BO13] for a discussion about this condition.)

We consider also the following two properties of FE schemes for Turing machines.

- *Succinctness:* A FE scheme for p-TM is said to be succinct if the ciphertexts have size polynomial in the security parameter and in the message size but independent on p (except for poly-logarithmic factors), and the tokens generated using KeyGen for machine M have size $q(\lambda, |M|)$, where q is a polynomial and $|M|$ is the size of the Turing machine, but independent on p (except for poly-logarithmic factors).
- *Input-specific running-time:* A FE scheme is said to have input-specific run time if the decryption algorithm on input token $\mathtt{Tok}$ for machine M and encryption of message m, takes time $p(\lambda, \mathsf{time}(M, m))$. Moreover, in this case the scheme is a for a different functionality (that also outputs the time of the computation) changed in the obvious way.

Indistinguishability-based security. The indistinguishability-based notion of security for functional encryption scheme
$\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, M) is formalized by means of the following game $\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}$ between an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* $\mathcal{C}$. Below, we present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

$\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda)$

1. $\mathcal{C}$ generates $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and runs $\mathcal{A}_0$ on input Pk;
2. $\mathcal{A}_0$ submits queries for keys $k_i \in K$ for $i = 1, \ldots, q_1$ and, for each such query, $\mathcal{C}$ computes $\mathtt{Tok}_i = \mathsf{KeyGen}(\mathsf{Msk}, k_i)$ and sends it to $\mathcal{A}_0$.

   When $\mathcal{A}_0$ stops, it outputs two *challenge plaintexts* $m_0, m_1 \in M$ satisfying $|m_0| = |m_1|$ and its internal state $\mathtt{st}$.
3. $\mathcal{C}$ picks $b \in \{0, 1\}$ at random, computes the *challenge ciphertext* $\mathsf{Ct} = \mathsf{Enc}(\mathsf{Pk}, m_b)$ and sends Ct to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$.
4. $\mathcal{A}_1$ submits queries for keys $k_i \in K$ for $i = q_1 + 1, \ldots, q$ and, for each such query, $\mathcal{C}$ computes $\mathtt{Tok}_i = \mathsf{KeyGen}(\mathsf{Msk}, k_i)$ and sends it to $\mathcal{A}_1$.
5. When $\mathcal{A}_1$ stops, it outputs b'.
6. **Output:** if $b = b'$, m_0 and m_1 are of the same length, and $F(k_i, m_0) = F(k_i, m_1)$ for $i = 1 \ldots, q$, then output 1 else output 0.

The advantage of adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}^{\mathsf{FE},\mathsf{IND}}_{\mathcal{A}}(1^\lambda) = |\mathrm{Prob}[\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda) = 1] - 1/2|.$$

Definition C.2 We say that FE is *indistinguishably secure* (IND-Ssecure, for short) if all probabilistic polynomial-time adversaries $\mathcal{A}$ have at most negligible advantage in the above game.

Simulation-based security In this section, we give a *simulation-based* security definition for FE similar to the one given by Boneh, Sahai and Waters [BSW11]. For simplicity of exposition, below, we present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

Definition C.3 [Simulation-Based security] A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, M) is *simulation-secure* (SIM security, for short) if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for all *adversary* algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ the outputs of the following two experiments are computationally indistinguishable.

$\mathsf{RealExp}^{\mathsf{FE},\mathcal{A}}(1^\lambda)$	$\mathsf{IdealExp}^{\mathsf{FE},\mathcal{A}}_{\mathsf{Sim}}(1^\lambda)$
$(\mathsf{Pk},\mathsf{Msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$;	$(\mathsf{Pk},\mathsf{Msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$;
$(m,\mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;	$(m,\mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk},m)$;	$(\mathsf{Ct},\mathtt{st}') \leftarrow \mathsf{Sim}_0(\mathsf{Pk},\lvert m\rvert,\{k_i,\mathtt{Tok}_{k_i},F(k_i,m)\},\tau)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk},\mathsf{Ct},\mathtt{st})$;	$\alpha \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot)}(\mathsf{Pk},\mathsf{Ct},\mathtt{st})$;
Output: (Pk,m,α)	**Output:** (Pk,m,α)

Here, $\{k_i\}$ correspond to the token queries of the adversary. Further, oracle $\mathcal{O}(\cdot)$ is the second stage of the simulator, namely algorithm $\mathsf{Sim}_1(\mathsf{Msk},\mathtt{st}',\cdot,\cdot)$. Algorithm Sim_1 receives as third argument a key k_j for which the adversary queries a token, and as fourth argument the output value $F(k_j,m)$. Further, note that the simulator algorithm Sim_1 is stateful in that after each invocation, it updates the state $\mathtt{st}'$ which is carried over to its next invocation.

Remark C.4 [Random Oracle Model] In the random oracle model, the output of the real experiment includes the queries made by any algorithm to the random oracle and the responses. In the ideal experiment, the simulator handles all such queries and the output of the experiment includes the queries and responses (given by the simulator) as well. This can be formalized as done in the context of Zero-Knowledge by Wee [Wee09].

Remark C.5 [(q_1,q_c,q_2)-SIM — Bounded messages and queries] To make the definition more precise, we define (q_1,q_c,q_2)-SIM security, where $q_1 = q_1(\lambda), q_c = q_c(\lambda), q_2 = q_2(\lambda)$ are polynomials that are fixed a priori, as follows. In this definition, we require that SIM-security as define above holds for adversaries $\mathcal{A}$ where $\mathcal{A}_0$ makes at most q_1 queries and outputs message vectors containing at most q_c messages, and furthermore $\mathcal{A}_1$ makes at most q_2 queries. In the case that a parameter is an unbounded polynomial we use the notation poly. Thus, for example, (q_1,q_c,poly)-SIM security means that the adversary in the security definition makes a q_1-bounded number of non-adaptive key derivation queries but an unbounded number of adaptive key-derivation queries, and outputs a q_c-bounded message vector. We call a FE scheme *bounded* if it achieves (q_1,q_c,q_2)-SIM-Security in which one of the parameters q_1,q_c,q_2 is not poly.

Remark C.6 [Simulated Setup] The above follows the security definition of [GVW12a] in that in the ideal experiment, the setup and non-adaptive key derivation queries are handled honestly (not by the simulator). More generally, we could have an algorithm $\mathsf{Sim.Setup}$ in place of $\mathsf{FE.Setup}$ in the ideal experiment; that is, the parameters of the ideal experiment could be generated by the simulator. In this setting the non-adaptive queries are handled by the simulator as well. We assume this slightly weaker security definition definition in Section Section 3.2.

Remark C.7 Actually, the syntax of a bounded FE scheme should take in account the parameters q_1,q_c,q_2, that is the setup should take as input such parameters but for simplicity we omit these details.

D MI-FE and its IND-Security

Let us define a multi-input functional encryption (MI-FE) scheme by means of an ℓ-ary functionality.

Definition D.1 [ℓ-ary Functionality] A *functionality* $F =$ is a family of functions $F : K \times X^\ell \to \Sigma$ where K is the *key space*, X is the *message space* and Σ is the *output space*.

In this work, our main focus will be on the following functionality.

Definition D.2 [p-TM_ℓ Functionality][14] The p-TM_l functionality for a polynomial $p()$ has key space K equals to the set of all l-input Turing Machines, which run time depends only on the inputs lengths $|m_1|, \ldots, |m_\ell|$ and equals $p(|m_1| + \cdots + |m_\ell|)$. Message space M is the set $\{0,1\}^*$. For $M \in K$ and $m_1, \ldots, m_l \in M$, we have $\mathsf{TM}_l(M, m_1, \ldots, m_l) = M(m_1, \ldots, m_l)$.

Remark D.3 We recall that the work of Goldwasser *et al.* [GGJS13] on MI-FE focuses on the circuit model but therein it is also sketched how to extend the results to the Turing Machine model. Similar considerations hold for the schemes of Gordon *et al.* [GKL$^+$13] as well. Further details will be given in the Master's Thesis of the second author. Moreover, as we point out several times in this work, for our results we do *not* need MI-FE that are IND-Secure but only CRIND-Secure ones (see Section 2.3) for which we will provide direct constructions.

Definition D.4 [Multi-Input Functional Encryption Scheme] A *multi-input functional encryption* (MI-FE, in short) scheme for ℓ-ary functionality F defined over (K, X) is a tuple $\mathsf{MI\text{-}FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ of 4 algorithms with the following syntax:

1. $\mathsf{Setup}(1^\lambda)$ outputs *master public key* Mpk consisting of ℓ encryption keys $\mathsf{Ek}_1, \ldots, \mathsf{Ek}_l$ and *master secret key* Msk for *security parameter* λ.

2. $\mathsf{KeyGen}(\mathsf{Msk}, k)$, on input a master secret key Msk and *key* $k \in K_n$ outputs *token* $\mathtt{Tok}$.

3. $\mathsf{Enc}(\mathsf{Ek}, x)$, on input encryption key $\mathsf{Ek} \in \mathsf{Mpk}$ and *message* $x \in X$ outputs *ciphertext* Ct.

   In the case where all of the encryption keys Ek_i are the same, we assume that each ciphertext Ct has an associated label i to denote that the encrypted plaintext constitutes an i-th input to the functionality. For convenience of notation, we omit the labels from the explicit description of the ciphertexts. In particular, note that when Ek_i's are distinct, the index of the encryption key Ek_i used to compute Ct implicitly denotes that the plaintext encrypted in Ct constitutes an i-th input to the functionality, and thus no explicit label is necessary.

4. $\mathsf{Eval}(\mathsf{Mpk}, \mathtt{Tok}_k, \mathsf{Ct}_1, \ldots, \mathsf{Ct}_\ell)$ on input master public key Mpk, secret key for key k and ℓ ciphertexts $\mathsf{Ct}_1, \ldots, \mathsf{Ct}_\ell$, outputs $y \in \Sigma \cup \{\perp\}$.

In addition we make the following *correctness* requirement: for all $k \in K$ and all $(x_1, \ldots, x_\ell) \in X^\ell$:

$$\Pr\left[\begin{array}{l}(\mathsf{Mpk} = (\mathsf{Ek}_1, \ldots, \mathsf{Ek}_\ell), \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda); \mathtt{Tok}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k); \\ \mathsf{Eval}(\mathsf{Mpk}, \mathtt{Tok}_k, \mathsf{Encrypt}(\mathsf{Ek}_1, x_1), \ldots, \mathsf{Encrypt}(\mathsf{Ek}_\ell, x_\ell)) \neq F(k, x_1, ..., x_\ell)\end{array}\right] = \mathsf{negl}(\lambda) \ ,$$

where the probability is taken over the coins of $\mathsf{Setup}, \mathsf{KeyGen}$ and $\mathsf{Encrypt}$.

We consider also the following two properties of MI-FE schemes for Turing machines.

[14] This functionality was only implicitly assumed in Goldwasser *et al.* [GGJS13] and other works but not formally defined.

- *Succinctness:* A MI-FE scheme for p-TM is said to be succinct if the ciphertexts have size polynomial in the security parameter and in the size of the messages but independent on p (except for poly-logarithmic factors), and the tokens generated using KeyGen for machine M have size $q(\lambda, |M|)$, where q is a polynomial and $|M|$ is the size of the Turing machine, but independent on p (except for poly-logarithmic factors).
- *Input-specific running-time:* A MI-FE scheme is said to have input-specific run time if the decryption algorithm on input token $\mathtt{Tok}$ for machine M and encryptions of messages $m_1, \ldots, m_\ell$, takes time $\mathsf{poly}(\lambda, \mathsf{time}(M, m_1, \ldots, m_\ell))$. Moreover, in this case the scheme is a for a different functionality (that also outputs the time of the computation) changed in the obvious way.

We point out that in this paper we do not need to assume IND-Secure MI-FE schemes but only CRIND-Secure ones (see Section 2.3). For completeness we recall the definition of IND-Security for MI-FE schemes.

IND-Security of MI-FE. The indistinguishability-based notion of security for a multi-input functional encryption scheme [GGG+14, GKL+13, GGJS13] $\mathsf{MI\text{-}FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for an ℓ-ary functionality F defined over (K, M^ℓ) is formalized by means of the following experiment $\mathsf{IND}_{\mathcal{A}}^{\mathsf{MI\text{-}FE}}$ with an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. Below, we present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

$\mathsf{IND}_{\mathcal{A}}^{\mathsf{MI\text{-}FE}}(1^\lambda)$

1. $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{MI\text{-}FE.Setup}(1^\lambda)$
2. $(X^0, X^1, \mathtt{st}_0) \leftarrow \mathcal{A}_0^{\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk},\cdot)}(1^\lambda, \mathsf{Pk})$ where $X^b = (x_1^b, \ldots, x_\ell^b)$ and we require that $|x_i^0| = |x_i^1|$ for every i;
3. $b \xleftarrow{R} \{0,1\}$;
4. $\mathsf{Ct}_i \leftarrow \mathsf{MI\text{-}FE.Encrypt}(\mathsf{Ek}_i, x_i^b)$ for $i \in \{1, \ldots, \ell\}$;
5. $b' \leftarrow \mathcal{A}_1^{\mathsf{MI\text{-}FE.KeyGen}(\mathsf{Msk},\cdot)}(\mathtt{st}_0, (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_\ell))$;
6. **Output:** $(b = b')$.

In above game we make following additional requirement:

- Let K denote the entire set of key queries made by adversary $\mathcal{A}$. Then, the challenge vectors X_0 and X_1 chosen by $\mathcal{A}_0$ must be compatible with K. This means that for every $i \in \{1, \ldots, \ell\}$ and $k \in K$, it holds that $F(k, \cdot, \ldots, \cdot, x_i^0, \cdot, \ldots, \cdot) = F(k, \cdot, \ldots, \cdot, x_i^1, \cdot, \ldots, \cdot)$.

The advantage of adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MI\text{-}FE},\mathsf{IND}}(1^\lambda) = |\mathrm{Prob}[\mathsf{IND}_{\mathcal{A}}^{\mathsf{MI\text{-}FE}}(1^\lambda) = 1] - 1/2|.$$

Definition D.5 We say that $\mathsf{MI\text{-}FE}$ is *indistinguishably secure* (IND security, for short) if all probabilistic polynomial-time adversaries $\mathcal{A}$ have at most negligible advantage in the above game.