On the Power of Rewinding Simulators in Functional Encryption

Angelo De Caro[1] Vincenzo Iovino[2]

[1] NTT Secure Platform Laboratories, Japan, Angelo.Decaro@lab.ntt.co.jp
[2] University of Warsaw, Poland, iovino@dia.unisa.it

Abstract

In the recent years, functional encryption (FE) has received a lot of attention due to its versatility and unique challenges it poses. In FE, a receiver with secret-key sk_y can compute from an encryption of x the value $F(y, x)$ for some functionality F. The seminal work of Boneh, Sahai and Waters [TCC'11] showed that for functional encryption the indistinguishability notion of security (IND-Security) is weaker then simulation-based and, moreover, showed that simulation-based security is impossible to achieve even in weaker settings. This has opened up the door to a plethora of papers, showing feasibility and new impossibility results, having in common the pursuit of a reasonable and achievable simulation-based security definition.

With the same aim, in this work, we propose a new simulation-based security definition that we call *rewinding simulation-based security* (RSIM-Security). Rewinding arguments have been used in all sorts of interactive protocols and have been shown to be highly useful to argue security. We exploit this power allowing the simulator to rewind the adversary under specific constraints. Specifically, the simulator will be able to rewind the adversary an arbitrary number of times under the constraint that the simulator does not learn more information about the challenge messages than the adversary.

Under our new definition we show that:

(1) IND-Security is equivalent to RSIM-Security for *predicate encryption with public-index* (i.e. Attribute-Based Encryption) in the *standard model*. Previous results showed impossibility results in the standard model.

This *equivalence* is the best one can hope for general functionalities due to the counterexample of Boneh *et al.*.

(2) Notwithstanding, we show that for specific (private-index) functionalities (e.g., Anonymous IBE, inner-product over $\mathbb{Z}_2$, any family of circuits in NC_0, and monotone conjunctive Boolean formulae) IND-Security is equivalent to RSIM-Security in the standard model. Previous results showed impossibility results in the standard model and the positive results were set either in the random oracle or in more restricted security definitions.

(3) On the negative side, we show that our security definition cannot be achieved by functional encryption schemes for general functionalities (specifically, functionalities that compute a pseudo-random function) in the adaptive setting. The argument we use is to some extent the *dual* of that used by Agrawal, Gorbunov, Vaikuntanathan, and Wee [CRYPTO'13] in the non-adaptive setting.

(4) We complete the picture showing the achievability of unbounded simulation (USIM) answering positively to a question posed by Agrawal, Gorbunov, Vaikuntanathan and Wee [CRYPTO 2013].

Keywords: Functional Encryption, Simulation-Based Security, Rewinding.

Contents

1 Introduction

Functional encryption (FE, for short) is a much more sophisticated type of encryption that was first proposed by Sahai and Waters in 2005 [SW05] and formalized by Boneh, Sahai and Waters in 2011, [BSW11]. Informally, in a functional encryption system, a decryption key allows a user to learn a *function* of the encrypted data. More specifically, in a functional encryption scheme 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 Mpk can generate a secret key Sk_k that allows the computation of $F(k, x)$ from a ciphertext of x computed under master public key Mpk. In other words, a functional encryption scheme generalizes classical encryption schemes where the unique secret key allows to compute the entire plaintext. A notable subclass of functional encryption is that of *predicate encryption* (PE, for short) which are defined for functionalities whose message space X consists of two subspaces I and M called respectively *index space* and *payload space*. In this case, the functionality F is defined in terms of a polynomial-time predicate $P : K \times I \to \{0, 1\}$ as follows: $F(k, (\mathsf{ind}, \mathsf{m})) = \mathsf{m}$ if $P(k, \mathsf{ind}) = 1$, $\perp$ otherwise, where $k \in K$, $\mathsf{ind} \in I$ and $\mathsf{m} \in M$. Those schemes are also called *predicate encryption with private-index*. On the other hand, when the index ind is easily readable from the ciphertext those schemes are called *predicate encryption with public-index* (PIPE, for short). Examples of such schemes are Identity-Based Encryption (IBE, for short) [Sha85, BF01, Coc01], Anonymous Identity-Based Encryption (AIBE, for short) [BF01, **?**], Attribute-Based Encryption (ABE, for short) [SW05, GPSW06], Functional Encryption for Regular Languages [Wat12], Inner-Product Encryption [BW07, KSW08, LOS$^+$10, OT12] and Functional Encryption for General Circuits [GVW12, GGH$^+$13a, BCP13, ABG$^+$13] among others.

Unlike in the case of classical cryptosystems, a general study of the security of functional encryption did not appear initially. Instead, progressively more expressive forms of FE were constructed in a series of works that adopted indistinguishability-based (IND) notions of security. The study of simulation-based (SIM) notions of security for functional encryption were initiated only comparatively recently by Boneh, Sahai, and Waters [BSW11] and O'Neill [O'N10] who explored security definitions for functional encryption that arise from the simulation paradigm [GM84, GMR85, GMW86] that has been showed to be really powerful in capturing the security in many contexts as the closely related of secure computation protocols. The aim of these simulation-based definitions was to capture the most basic intuition about security for FE, namely that getting the secret key Sk_k corresponding to the key $k \in K$ should only reveal $F(k, x)$ when given an encryption of x. Quite interestingly, they show there exist clearly insecure FE schemes for certain functionalities that are nonetheless provably IND-Secure, whereas these schemes do not meet the stronger notion of SIM-Security. On the other hand, negative and positive results have also emerged showing that SIM-Security is achievable [BSW11, O'N10, BO12, GVW12, AGVW13, DIJ$^+$13] only in specific settings.

So far, all the simulation-based security definitions for functional encryption known in the literature share a common characteristic. They all require the simulator to have the same key-generation query distribution of the adversary. This requirements is natural in that it requires that the simulator does not gain more information than the adversary. On the other hand it results in essentially straight-line simulators that cannot rewind the adversary (In particular, in the definitions used by [GVW12, AGVW13, DIJ$^+$13] the simulator is defined directly to be straight-line). By rewinding, we mean that the simulator runs parts of the adversary during the simulation and produces a fragment of the conversation that has some desired property with a certain probability. If the simulator fails then it possibly gains some information useful to produce a successful simulation and then rewinds, that is it just runs the part of the adversary

again from scratch. Rewinding has been used in all sorts of interactive protocols and has been showed to be highly useful to argue security. This leads to the main question that we study in this work:

> *Is there a way to define simulation-based security for FE allowing the simulator to rewind the adversary in such a way the simulator does not gain more information than the adversary? To what extent are rewinding simulators useful to overcome the known impossibility results for SIM-Secure functional encryption?*

1.1 Previous Works

	Public-Index	AIBE	All circuits
$(\mathsf{poly},\mathsf{poly},\mathsf{poly})$-IND	**yes** [GVW13, GGH$^+$13b]	**yes**	**yes** [GGH$^+$13a, ABG$^+$13, BCP13]
$(\mathsf{poly},\mathsf{poly},\mathsf{poly})$-SIM	**yes** [BSW11] (RO)	**yes** [BSW11] (RO)	**no** [AGVW13, BSW11, BO12]
$(q_1,0,1)$-SIM	**yes** $\uparrow$	**yes** $\uparrow$	**yes** [GVW12]
$(q_1,0,\mathsf{poly})$-SIM	**yes** $\uparrow$	**yes** $\uparrow$	**yes** [GKP$^+$13]
(q_1,poly,ℓ)-SIM	**yes** $\uparrow$	**yes** $\uparrow$	**yes** [DIJ$^+$13]

Table 1: Summary of the previous results. Results implied by results in the previous row are marked with $\uparrow$. The first column indicates the security definition. The notion of IND-Security is presented in Definition 2.3 and for the notion of SIM-Security it is enough to consider that of [DIJ$^+$13] (For the reader convenience we report the [DIJ$^+$13]'s definition in Appendix B). The second, third and fourth columns indicate respectively whether the definition is achievable for public-index predicate encryption (i.e., ABE), Anonymous Identity-based Encryption and functional encryption for poly-size circuits. RO is the random oracle model.

Results about functional encryption now live in a high-dimensional space, where there are many parameters and several results ruling out or allowing schemes for certain parameters. Before presenting these results, summarized in Table 1, to make things clear, following [DIJ$^+$13] notation, we define (q_1,q_2,ℓ)-atk-Security, where $q_1 = q_1(\lambda), q_2 = q_2(\lambda), \ell = \ell(\lambda)$ are polynomials in the security parameter λ that are fixed a priori and $\mathsf{atk} \in \{\mathsf{IND},\mathsf{SIM}\}$, as follows. Specifically, atk-Security holds for adversaries $\mathcal{A}$ that issues at most q_1 *non-adaptive* key-generation queries, output *challenge message vectors* of length at most ℓ, and furthermore issues at most q_2 *adaptive* key-generation queries. In the case that a parameter is an unbounded polynomial we use the notation poly. Thus, for example, if q_1 and ℓ are polynomials then (q_1,poly,ℓ)-SIM-Security means that the adversary in the SIM-Security definition makes a $q_1(\lambda)$-bounded number of non-adaptive key-generation queries but an unbounded number of adaptive key-generation queries, and outputs a $\ell(\lambda)$-bounded challenge message vector, where λ is the security parameter. If the parameters are not specified we intend them set to poly. (IND-Security is defined in Definition 2.3. As reference for SIM-Security, we take the definition of [DIJ$^+$13] which we report, for reader convinience, in Appendix B together with the [BSW11] definition.)

In the seminal work of Boneh, Sahai and Waters [BSW11], it was shown that for FE, unlike classical encryption, IND-Security is weaker than SIM-Security. Indeed, the authors show a clearly insecure FE scheme that is provably IND-Secure. Moreover, in the same work Boneh *et al.* show that $(0,2,\mathsf{poly})$-SIM-Security is *impossible* to achieve even for a simple functionality like IBE in the *non-programmable oracle model*, but prove, in the random oracle model, that $(\mathsf{poly},\mathsf{poly},\mathsf{poly})$-$\mathsf{IND}$-Security is equivalent to $(\mathsf{poly},\mathsf{poly},\mathsf{poly})$-$\mathsf{SIM}$-Security for predicate encryption with public-index, and there exists an AIBE scheme that is $(\mathsf{poly},\mathsf{poly},\mathsf{poly})$-$\mathsf{SIM}$-Security. At the same time, O'Neill [O'N10] does similar considerations and shows that for

pre-image sampleable functionalities, $(\mathsf{poly}, 0, \mathsf{poly})$-IND-Secuirty is equivalent to $(\mathsf{poly}, 0, \mathsf{poly})$-SIM-Secure. Later, Bellare and O'Neill [BO12] show that the impossibility result of [BSW11] also extends to the standard model assuming the existence of collision resistant hash functions. Furthermore, new definitions were introduced to the aim of overcoming the impossibility results. Specifically, they define a new notion equivalent to IND-Security and thus incurring in the same deficiency, and a new simulation-based definition for which a proof of security was only shown for functionalities with key space of polynomial size (and so not including basic functionalities like IBE). In 2012, Gorbunov *et al.* [GVW12] presented a construction of FE for general circuits that is $(q_1, 0, \mathsf{poly})$-SIM-Secure. Later, Agrawal *et al.* [AGVW13] proved an impossibility result showing that it is *impossible* to achieve $(\mathsf{poly}, 1, 0)$-SIM-Security. Furthermore, in the same paper, the authors prove that $(\mathsf{poly}, \mathsf{poly}, 1)$-IND-Security implies $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-IND-Security, and propose a simulation-based notion of security that considers computational *unbounded* simulator as a way to overcome current impossibility results, leaving many open problems about this definition. Then, Goldwasser *et al.* [GKP$^+$13] presented an FE for general circuits with succinct ciphertexts, meaning that the size of the ciphertext does grow only with the respect of the depth of the circuits to be evaluated, provable $(q_1, 0, \mathsf{poly})$-SIM-secure. In 2013, De Caro *et al.* [DIJ$^+$13] presented a general compiler to transform any $(q_1, \mathsf{poly}, \ell)$-IND-Secure FE scheme for circuits into one that is $(q_1, \mathsf{poly}, \ell)$-SIM-Secure matching the known impossibility results. Finally, in recent breakthroughs, Gorbunov *et al.* and Garg *et al.* [GVW13, GGH$^+$13b] proposed $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-IND-Secure constructions for predicate encryption with public-index for general circuits, and [GGH$^+$13a, ABG$^+$13, BCP13] proposed the first candidate constructions for a $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-IND-Secure[1] functional encryption scheme for general circuits from indistinguishable obfuscation and extractable obfuscation.

1.2 Our main results

In Section 3, we propose a new notion of simulation-based security that we call *rewinding simulation-based security* (RSIM-Security, for short). RSIM-Security strengthens the rewinding power of the simulator. Specifically, the simulator will be able to rewind the adversary an arbitrary number of times under the constraints that the simulator does not learn more information about the challenge messages than the adversary. This is obtained by forbidding the simulator to access the oracle of the functionality directly. When the adversary invokes its key-generation oracle on input a key k then the simulator gets in input the key k and $F(k, x)$, where x is the challenge message, and is asked to generate the secret key for k.

The first consideration about our definition is that the example used by [BSW11, BO12] to show that $(0, 2, \mathsf{poly})$-SIM-Secure IBE is impossible to achieve does not apply to our definition. (We recall the [BSW11] definition in Appendix B.) Informally, let us consider the following adversary $\mathsf{Adv} = (\mathsf{Adv}_0, \mathsf{Adv}_1)$ for the IBE functionality. Adv_0 returns as challenge messages the vector $((0, r_i))_{i \in [\ell]}$, where $\ell = kl(\lambda) + \lambda$, kl is a polynomial bounding the length of secret keys, 0 is the identity and r_i is a random bit for each $i \in [\ell]$. Then, Adv_1 invokes its key-generation oracle on input identity $w = \mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1|| \cdots ||\mathsf{Ct}_\ell)$ for some collision-resistant hash function CRHF, and then asks to see a secret key for identity 0. The output of Adv_1 is the transcript of its entire computation including its inputs. The strategy of this adversary is not successful with the respect to our security definition because once the RSIM simulator gets the r_i's (this happens when the adversary issues key-generation query for identity 0), then the simulator can simply rewind the adversary on input new simulated ciphertexts tailored on the r_i's. Notice,

[1] Precisely, the functional encryption scheme of [GGH$^+$13a] only satisfies selective $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-IND-Security but later [BCP13] and [ABG$^+$13] provided schemes that achieve full security.

that the simulator has just learnt what the adversary is learning about the challenge messages.

Unfortunately, the above does not mean that our definition is achievable in general but it opens up to the following positive and negative results that are summarized in Table 2.

Positive results. In Section 4, we show that for predicate encryption with public-index $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-IND-Secure is equivalent to $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-RSIM-Secure in the standard model, thus closing this open problem left by [BSW11, AGVW13]. We complement this result by showing that it is the best one can hope for. Namely, in Section 5, adapting the counterexample of [BSW11], we show a predicate for which is possible to construct a predicate encryption with private-index scheme that reveals more information than what it should be, but nevertheless it is IND-secure. Of course, the scheme is not RSIM-Secure. However, in Section 6 we generalize the equivalence result for predicate encryption with public-index to predicate encryption with private-index for specific functionalities, namely Anonymous IBE, inner-product over $\mathbb{Z}_2$, any family of circuits in NC_0, and monotone conjunctive Boolean formulae. Finally, in Section 6.4 we prove that the brute-force construction of [BW07, BSW11] is $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-RSIM-Secure in the standard model assuming only the IND-CPA security of the underlying public-key encryption scheme whereas [BSW11] showed that a slightly modified variant of the [BW07] scheme can be proven $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-SIM-Secure in the random oracle, and [BO12] proved its $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-SIM-Security in the standard model assuming that the underlying PKE scheme is also secure against selective opening key attack.

	Public-Index	Private-Index	All circuits
$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-RSIM	**yes** (Section 4)	**yes** (Section 6)	**no** [AGVW13], (Section 7)
$(\mathsf{poly}, 0, 1)$-RSIM	**yes** $\uparrow$	**yes** $\uparrow$	**no** [AGVW13]
$(0, \mathsf{poly}, \mathsf{poly})$-RSIM	**yes** $\uparrow$	**yes** $\uparrow$	**no** (Section 7)
$(\mathsf{poly}, 0, \mathsf{poly})$-USIM	**yes** $\rightarrow$	**yes** $\rightarrow$	**yes** (Appendix C)

Table 2: Summary of our results. Results implied by results in the next column are marked with $\rightarrow$. All the results are in the standard model. Results implied by results in the previous row are marked with $\uparrow$. The first column indicates the security definition. The notion of RSIM-Security is defined by Definition 3.1. The second, third and fourth columns indicate respectively whether the definition is achievable for public-index predicate encryption (in this case, we support *any* predicate), predicate encryption for specific predicates (see Section 6 for more details on the predicates supported that include AIBE), and functional encryption for circuits. The impossibility result of Section 7 for $(0, 1, \mathsf{poly})$-RSIM-Security is for the auxiliary input setting. For simplicity the latter result is stated in the table for $(0, \mathsf{poly}, \mathsf{poly})$-RSIM-Security but also holds for $(0, 1, \mathsf{poly})$-RSIM-Security. Our positive results are based on the assumption of the existence of IND-Secure schemes as the recent schemes of [GGH$^+$13a, ABG$^+$13, BCP13].

Impossibility for General Circuits. Unfortunately, the impossibility results of [AGVW13] still holds with respect to our definition. Interestingly, in Section 7, we extend their impossibility result to show that $(0, 1, \mathsf{poly})$-RSIM-Security cannot be achieved by functionalities that compute a pseudo-random function.[2] Our result, as is the case with those of [BSW11, BO12], is a trade-off. It shows that RSIM-Security requires long secret keys, meaning that the total number of

[2]Precisely, this result is for the auxiliary input setting (see Section 3). The auxiliary input setting has been already used by [BO12] in the same context.

bits in messages securely encrypted must be bounded by the length of a secret key. The main difference with the previous results is that for RSIM-Security this trade-off shows up only when considering richer class of functionality containing at least pseudo-random functions. In this sense, our result can be seen as the *dual* of that of [AGVW13].

Unbounded Simulation. Finally, in Appendix C we complete the picture by showing the achievability of unbounded simulation (USIM-Security) answering positively to a question open by [AGVW13]. A similar fact has been noticed by [BR13] in the context of obfuscation.

2 Definitions

Notation. A *negligible* function $\mathsf{neg}(\lambda)$ is a function that is smaller than the inverse of any polynomial in λ. If x_1 and x_2 are binary strings, we denote by $x_1||x_2$ or (x_1, x_2) their concatenation. We denote by $[n]$ the set $\{1, \ldots, n\}$. If x is a binary string we denote by $|x|$ the bit length of x, we denote by x_i the i-th bit of x, $1 \leq i \leq |x|$. PPT is a shorthand for Probabilistic Polynomial-Time. We denote by $A(x; r)$ the execution of a PPT algorithm A with input x and randomness r. Sometimes we simply write $A(x)$ instead of $A(x; r)$ when it is clear from the context. If B is an algorithm and A is an algorithm with access to an oracle then $A^B(\cdot)$ denotes the execution of A with oracle access to $B(\cdot)$.

Following Boneh *et al.* [BSW11], we start by defining the notion of functionality.

Definition 2.1 [Functionality] A *functionality* F defined over (K, X) is a function $F : K \times X \to \Sigma \cup \{\perp\}$ where K is the *key space*, X is the *message space* and Σ is the *output space* and $\perp$ is a special string not contained in Σ. Notice that the functionality is undefined for when either the key is not in the key space or the message is not in the message space. Furthermore we require that there are efficient procedures to check membership of a string in the message space and key space and to sample from these spaces.

A functional encryption scheme FE for functionality F is defined as follows.

Definition 2.2 [Functional Encryption Scheme] A *functional encryption* (FE) 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 $(\mathsf{Mpk}, \mathsf{Msk})$ for *security parameter* λ.
2. $\mathsf{KeyGen}(\mathsf{Msk}, k)$, on input a master secret key Msk and *key* $k \in K$ outputs *secret key* Sk_k.
3. $\mathsf{Enc}(\mathsf{Mpk}, x)$, on input public key Mpk and *message* $x \in X$ outputs *ciphertext* Ct;
4. $\mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}_k)$ outputs $y \in \Sigma \cup \{\perp\}$.

In addition we make the following *correctness* requirement: for all $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, all $k \in K_n$ and $m \in M_n$, for $\mathsf{Sk} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, m)$, we have that $\mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}) = F(k, m)$ whenever $F(k, m) \neq \perp$[3], except with negligible probability.

[3]See [BO12, ABN10] for a discussion about this condition.

The empty key. For any functionality, we also assume that the key space contains a special *empty key* ϵ such that $F(\epsilon, x)$ gives the length of x and (depending on the functionality) some intentionally leaked information on x that can be easily extracted from an encryption of x. When $\vec{x} = (x_1, \ldots, x_\ell)$ is a vector of messages, for any $k \in K \cup \{\epsilon\}$, we denote by $F(k, \vec{x})$ the vector of evaluations $(F(k, x_1), \ldots, F(k, x_\ell))$.

Further parametrizations. In general, the key space, the message space and the functionality itself are families of sets and functions indexed by the security parameter $\lambda \in \mathbb{N}$. Specifically, a functionality F is a family of functions $F = \{F_\lambda : K_\lambda \times X_\lambda \to \Sigma_\lambda \cup \{\perp\}\}_\lambda$ where $\{K_\lambda\}_\lambda$ is the *key space family*, $\{X_\lambda\}_\lambda$ is the *message space family* and $\{\Sigma_\lambda\}_\lambda$ is the *output space family*. It will be clear from the context which kind of formulation of functionality we adopt, whether for families or not. Thus, if functionality F is actually a family of functions, with a slight abuse of notation we will denote by $F(k, x)$ the value $F_\lambda(k, x)$, where λ is the security parameter.

Secret-key length. We say that a functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ has secret-key length $kl(\cdot)$ if $|\mathsf{Sk}| \leq kl(\lambda)$ for all $k \in K_\lambda$, $X \in X_\lambda$, all $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and all $\mathsf{Sk} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$. Note that every FE scheme must have some polynomial $kl(\cdot)$ secret-key length in order to be efficient.

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, X) is formalized by means of the following game $\mathsf{IND}^{\mathsf{FE}}_{\mathsf{Adv}}$ between an adversary $\mathsf{Adv} = (\mathsf{Adv}_0, \mathsf{Adv}_1)$ and a *challenger* $\mathcal{C}$.

1. $\mathcal{C}$ generates $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and runs Adv_0 on input Mpk;
2. Adv_0, during its computation, issues q_1 *non-adaptive key-generation queries.* $\mathcal{C}$ on input key $k \in K$ computes $\mathsf{Sk} = \mathsf{KeyGen}(\mathsf{Msk}, k)$ and sends it to Adv_0.

   When Adv_0 stops, it outputs two *challenge messages vectors*, of length ℓ, $\vec{x}_0, \vec{x}_1 \in X^\ell$ and its internal state `st`.
3. $\mathcal{C}$ picks $b \in \{0, 1\}$ at random, and, for $i \in \ell$, computes the *challenge ciphertexts* $\mathsf{Ct}_i = \mathsf{Enc}(\mathsf{Mpk}, x_b[i])$. Then $\mathcal{C}$ sends $(\mathsf{Ct}_i)_{i \in [\ell]}$ to Adv_1 that resumes its computation from state `st`.
4. Adv_1, during its computation, issues q_2 *adaptive key-generation queries.* $\mathcal{C}$ on input key $k \in K$ computes $\mathsf{Sk} = \mathsf{KeyGen}(\mathsf{Msk}, k)$ and sends it to Adv_1.
5. When Adv_1 stops, it outputs b'.
6. **Output:** if $b = b'$, $F(\epsilon, \vec{x}_0) = F(\epsilon, \vec{x}_1)$, and $F(k, \vec{x}_0) = F(k, \vec{x}_1)$ for each k for which Adv has issued a key-generation query, then output 1 else output 0.

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

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

Definition 2.3 We say that FE is (q_1, q_2, ℓ)-*indistinguishably secure* ((q_1, q_2, ℓ)-IND-Secure, for short) where $q_1 = q_1(\lambda), q_2 = q_2(\lambda), \ell = \ell(\lambda)$ are polynomials in the security parameter λ that

are fixed a priori, if all probabilistic polynomial-time adversaries Adv issuing at most q_1 non-adaptive key queries, q_2 adaptive key queries and output challenge message vectors of length and most ℓ, have at most negligible advantage in the above game. Notice that, in the case that a parameter is an unbounded polynomial we use the notation poly.

Predicate Encryption. A notable subclass of functional encryption is that of *predicate encryption* (PE, for short). Specifically, those schemes are defined for functionalities whose message space X consists of two subspaces I and M called respectively *index space* and *payload space*. In this case, the functionality F is defined in terms of a polynomial-time predicate $P : K \times I \to \{0,1\}$ as follows: $F(k,(\mathsf{ind},\mathsf{m})) = \mathsf{m}$ if $P(k,\mathsf{ind}) = 1$, $\perp$ otherwise, where $k \in K$, $\mathsf{ind} \in I$ and $\mathsf{m} \in M$. In particular, for the ϵ key, we have $F(\epsilon,(\mathsf{ind},\mathsf{m})) = (|\mathsf{ind}|,|\mathsf{m}|)$. As for general functionalities, a predicate P can be a family of predicates and in this case the functionality F defined in terms of P is a family of functions. Indistinguishable Security for PE is defined analogously to Definition 2.3.

Anonymous IBE. Let the key space $K_n = \{0,1\}^n$, index space $I_n = \{0,1\}^n$ and payload space $M_n = \{0,1\}^n$ the payload space for $n \in \mathbb{N}$. The predicate family $\mathsf{IBE} = \{\mathsf{IBE}_n : K_n \times I_n \to \{0,1\}\}_{n\in\mathbb{N}}$ is defined so that for any $k \in K_n, \mathsf{ind} \in I_n$, $\mathsf{IBE}(k,\mathsf{ind}) = 1$ if and only if $k = \mathsf{ind}$. We call a predicate encryption scheme (with private-index) for this predicate *Anonymous IBE* (AIBE, for short).

Predicate Encryption with Public-Index. It is a variant of predicate encryption (PIPE, for short) that makes the index easily readable from the ciphertext. In particular, in this type of FE the empty key ϵ explicitly reveals the index ind, namely $F(\epsilon,(\mathsf{ind},\mathsf{m})) = (\mathsf{ind},|\mathsf{m}|)$. Indistinguishable security is defined again analogously to Definition 2.3, with the main difference being in the adversary's challenge, namely it consists of two *payloads* $\mathsf{m}_0,\mathsf{m}_1$ and an index ind. An example of PIPE is *Identity-based Encryption* that is a PIPE for the predicate IBE defined before.

3 Rewinding Simulation-based Security

In this section, we present our definition of *rewinding simulation-based security*.

Definition 3.1 [Rewinding Simulation-based Security] A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup},\mathsf{KeyGen},\mathsf{Enc},\mathsf{Eval})$ for functionality F defined over (K,X) is (q_1,q_2,ℓ)-*rewinding simulation-secure* ((q_1,q_2,ℓ)-RSIM-Secure, for short), where $q_1 = q_1(\lambda), q_2 = q_2(\lambda), \ell = \ell(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0,\mathsf{Sim}_1)$ such that for all PPT *adversary* algorithms $\mathsf{Adv} = (\mathsf{Adv}_0,\mathsf{Adv}_1)$, issuing at most q_1 non-adaptive key queries, q_2 adaptive key queries and output challenge message vector of length and most ℓ, the outputs of the following two experiments are computationally indistinguishable. (Notice that, in the case that a parameter is an unbounded polynomial we use the notation poly.)

$\mathsf{RealExp}^{\mathsf{FE},\mathsf{Adv}}(1^\lambda)$

$(\mathsf{Mpk},\mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\vec{x},\mathtt{st}) \leftarrow \mathsf{Adv}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk})$;
$(\mathsf{Ct}_i \leftarrow \mathsf{Enc}(\mathsf{Mpk},x[i]))_{i\in\ell}$;
$\alpha \leftarrow \mathsf{Adv}_1^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk},(\mathsf{Ct}_i),\mathtt{st})$;
Output: $(\mathsf{Mpk},\vec{x},\alpha)$

$\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathsf{Adv}}(1^\lambda)$

$(\mathsf{Mpk},\mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\vec{x},\mathtt{st}) \leftarrow \mathsf{Adv}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk})$;
Let $\mathcal{Q} = (k_i, \mathsf{Sk}_{k_i}, F(k_i,\vec{x}))_{i\in[q_1]}$.
$\alpha \leftarrow \mathsf{Sim}_0^{\mathsf{Adv}_1^{\mathcal{O}}(\mathsf{Mpk},\cdot,\mathtt{st})}(\mathsf{Mpk},\mathsf{Msk},\mathcal{Q},F(\epsilon,\vec{x}))$;
Output: $(\mathsf{Mpk},\vec{x},\alpha)$

Here, the $(k_i \in K)_{i\in[q_1]}$'s are the q_1 keys for which Adv_0 has issued a *non-adaptive* key-generation query to its key-generation oracle, $F(k_i,\vec{x}) = (F(k_i,x[1]),\ldots,F(k_i,x[\ell]))$ and $F(\epsilon,\vec{x}) = (F(\epsilon,x[1]),\ldots,F(\epsilon,x[\ell]))$. In the ideal experiment Adv_1 is provided with a special oracle $\mathcal{O}$ for adaptive key-generation queries. The oracle $\mathcal{O}$ takes in input a key $k \in K$ and answers the query in the following way. The oracle invokes the simulator Sim_1 on input $(k,F(k,\vec{x}))$. Sim_1 outputs a secret key for k that the oracle returns to Adv_1. We require the simulator $\mathsf{Sim} = (\mathsf{Sim}_0,\mathsf{Sim}_1)$ to be stateful and allow Sim_0 and Sim_1 to communicate by means of a shared memory. We remark that each time Sim_0 runs the adversary Adv_1 on some input (Ct_i), Adv_1 is executed with input $(\mathsf{Mpk},(\mathsf{Ct}_i),\mathtt{st})$ and *fresh* randomness.

RSIM-Security $\implies$ IND-Security. We show this in Appendix D.

Auxiliary Inputs. Our definition can be extended naturally to the auxiliary input setting, as in Bellare and O'Neill [BO12]. An auxiliary input generator algorithm Z outputs z which is given to the adversary and simulator, and included in the output distribution of security game. Notice that, the simulator is not allowed to pick z. As in [BO12], the auxiliary input setting will be used in our impossibility result in Section 7, where z will contain a key for a collision-resistant hash function.

Relations with the Boneh *et al.* [BSW11] Definition. To clarify the difference with the Boneh *et al.* [BSW11] definition (See Appendix B for the formal definition), consider a simulator that performs the following simulation: (1) *First run.* Suppose that the adversary always issues one key-generation query after seeing the challenge ciphertext and let m be the challenge message output by the adversary (that the simulator does not know). In the first run, the simulator executes the adversary with input the encryption of m^0, and then the adversary issues key-generation query for key A. Now the simulator does not know how to continue the simulation. This can happen because $F(A,m) \neq F(A,m)$. Anyway, the simulator has possibly learnt some information. (2) *Second run.* At this point, the simulator rewinds the adversary with input the encryption of some message m^1, then the adversary issues key-generation query for key A_2 that the simulator answers correctly (using the master secret key). The simulator can answer this query and continue the simulation because, for example, $F(A_2,m) = F(A_2,m^1)$ and thus the view of the adversary during this run is indistinguishable from that in the real game (under the assumption of IND-Security of the underlying scheme). Finally, the adversary halts and simulator outputs what the adversary outputs.

Observe that the BSW definition forbids this kind of simulation since (1) the simulator is given direct access to the functionality oracle and (2) the key-generation queries issued by the simulator are given as input to the distinguisher. So according to the BSW definition, the

distinguisher would see 2 key-generation queries, and thus it could tell apart the real experiment where the adversary always asks one secret key from the ideal experiment.

On the other hand, in our definition the distinguisher would only see the *last* transcript. This is achieved by giving the simulator access to the functionality oracle only through the adversary. Thus, the simulator does not learn more information about the challenge messages than the adversary. Notice also, that the adversary could output the transcript of its whole computation that includes its key-generation queries but in this case the simulated transcript would contain an indistinguishable view anyway.

The definitions of [AGVW13, DIJ+13] also forbid this kind of simulation since their simulator is straight-line.

As we will show in the next sections, it is possible to exploit the power of the rewinding to implement simulators that can overcome current impossibility results for the previous definitions. To give a taste of how it could be done, recall the previous example. At the end of the first run, the simulator could have learnt some information on the hidden message m, for example the first bit of m, and thus m^1 has such bit set correctly and the new secret key for A_2 is 'coherent' with the simulated ciphertext (for example, it could be that the keys A and A_2 only depend on the first bit of the message).

4 An Equivalence for Public-Index Schemes

The main theorem of this section is the following.

Theorem 4.1 Let PIPE be an $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-$\mathsf{IND\text{-}Secure}$ PE scheme with public-index for predicate $P : K \times I \to \{0, 1\}$. Then, PIPE is $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-$\mathsf{RSIM}$-Secure as well.

Proof: Our simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ works as follows. Sim_0 takes in input the master public and secret key, the list $\mathcal{Q} = (k_i, \mathsf{Sk}_{k_i}, F(k_i, \vec{x}))_{i \in [q_1]}$, and the intentionally leaked information about the challenge messages $F(\epsilon, \vec{x}) = (\mathsf{ind}_j, |\mathsf{m}_j|)_{j \in [\ell]}$. Then, for each $i \in [q_1]$, Sim_0 checks whatever $P(k_i, \mathsf{ind}_j) = 1$ for some $j \in [\ell]$. If it is the case, then Sim_0 learns m_j. Furthermore, let $\mathcal{X}$ the tuple of messages (indices with the relative payloads) learnt by Sim_0. Then, for each pair in $\mathcal{X}$, Sim_0 generates a normal ciphertext by invoking the encryption algorithm. For all the other indices for which Sim_0 was not able to learn the corresponding payload, Sim_0 generates ciphertexts for those indices having a random payload. Let $\vec{x}^\star$ be the resulting message vector that the simulator used to produce the challenge ciphertexts.

Then, Sim_0 executes Adv_1 on input the so generated challenge ciphertexts. When Adv_1 invokes its key-generation oracle on input key k, Sim_1 is asked to generate a corresponding secret key given k and $F(k, \vec{x})$. Now we have two cases:

1. $P(k, \mathsf{ind}_j) = 1$ for some $j \in [\ell]$: Then, Sim learns m_j. If m_j was already known by Sim, it means that the corresponding challenge ciphertext was well formed when Sim invoked Adv_1. Then, Sim generates the secret key for k (using the master secret key) and continues the simulation. On the other hand, if Sim didn't know m_j then the ciphertext corresponding to ind_j was for a random message. Therefore, Sim must halt Adv_1. Sim adds $(\mathsf{ind}_j, \mathsf{m}_j)$ to $\mathcal{X}$ (and thus updates $\vec{x}^\star$) and with this new knowledge Sim rewinds Adv_1 on input the encryption of the new ciphertexts (i.e., the encryption of the new $\vec{x}^\star$).

2. $P(k, \mathsf{ind}_j) = 0$ for all $j \in [\ell]$: In this case, a secret key for k cannot be used to decrypt any of the challenge ciphertexts. Then, Sim generates the secret key for k (using the master secret key) and continues the simulation.

If at some point the adversary halts giving some output the simulator outputs what the adversary outputs. This conclude the description of the simulator Sim.

It remains to show that the simulated challenge ciphertexts does not change Adv_1's behaviour significantly. We call a key-generation query *good* if the simulator can answer such query without rewinding the adversary according to the previous rules. We call the execution of the simulator between two rewinds of the adversary a *run*. First, notice that the number of runs, meaning the number of times the simulator rewinds, is upper-bounded by the number of challenge messages ℓ that is polynomial in the security parameter. In fact, each time that a query is not good and the simulator needs to rewind then the simulator learn a new pair $(\mathsf{ind}_j, \mathsf{m}_j)$, for some $j \in [\ell]$ and the same query will never cause a rewind anymore. In the last run, that in which all the key-generation queries are good, the view of the adversary is indistinguishable from that in the real game. This follows from the IND-Security of PIPE. In fact, the evaluations of the secret keys on the challenge ciphertexts in the real experiment give the same values than the evaluation of the simulated secret keys on the simulated ciphertexts in the ideal experiment since the secret keys are generated honestly. Therefore, the IND-Security guarantees that in this case the view in the real experiment is indistinguishable from that in the ideal experiment. ∎

5 Insufficiency of Indistinguishable-Based Security for PE

In this section, we review the insufficiency for IND-Security shown by [BSW11]. Precisely, we show that the *equivalence* between indistinguishable-based security and simulation-based security in the standard model for *public-index predicate encryption* is the best one can hope for.

Namely, we will show a complex predicate for which is possible to construct a *private-index predicate encryption* scheme that is IND-Secure, but it is clearly insecure in practice. For completeness, we slightly adapt the example of [BSW11] extend it to a PE scheme (with private-index) that also encrypts a payload message. Specifically, let π be a *one-way permutation* and consider the predicate P defined as follows: $P(k, \mathsf{ind}) = 1$ if $k = \pi(\mathsf{ind})$, 0 otherwise. Now, consider the following implementation based on an Anonymous IBE scheme AIBE. The ciphertext for the pair $(\mathsf{ind}, \mathsf{m})$ is an AIBE's ciphertext for identity $\pi(\mathsf{ind})$ and payload $\mathsf{m}||\mathsf{ind}$, namely $\mathsf{AIBE.Enc}(\mathsf{AIBE.Mpk}, \pi(\mathsf{ind}), \mathsf{m}||\mathsf{ind})$. and a secret key for k is an AIBE's secret key for identity k, namely $\mathsf{AIBE.KeyGen}(\mathsf{AIBE.Msk}, k)$.

Clearly, given ciphertext for pair $(\mathsf{ind}, \mathsf{m})$ and secret key for k such that $\pi(\mathsf{ind}) = k$ reveal more information than need about the index ind.

Nevertheless, the new scheme can be proved IND-Secure. In fact, according to the constraints of the IND-Security, for challenge indices $\mathsf{ind}_0, \mathsf{ind}_1$, $P(k, \mathsf{ind}_0) = P(k, \mathsf{ind}_1) = 1$ if and only if $\pi(\mathsf{ind}_0) = k$ and $\pi(\mathsf{ind}_1) = k$ but this happens if and only if $\mathsf{ind}_0 = \mathsf{ind}_1$. Thus, the adversary can only issue secret key queries for k such that $P(k, \mathsf{ind}_0) = P(k, \mathsf{ind}_1) = 0$. Under this constraint it is easy to reduce the IND-Security of the new scheme to that of the AIBE scheme. On the other hand, the new scheme is not SIM-Secure (with the respect to any simulation-based security definition for FE known in the literature). Specifically, consider an adversary for the SIM-Security that chooses as challenge a pair $(\mathsf{ind}, \mathsf{m})$ for random index ind and random message m, and then asks the secret key for $k = \pi(\mathsf{ind})$, decrypts the challenge ciphertext using this secret key and outputs a transcript of its computation. The simulator has to generate an indistinguishable transcript only given as input $F(\epsilon, (\mathsf{ind}, \mathsf{m})) = (|\mathsf{ind}|, |\mathsf{m}|)$, $k = \pi(\mathsf{ind})$, the secret for k and the evaluation of the functionality on the challenge message, that in this case is just m being $\pi(\mathsf{ind}) = k$. Since ind is chosen at random and independently from the view of

the simulator, the probability of successful simulation reduces to inverting π.

6 Positive Results for PE with Private-Index

We have seen in the previous section that we cannot hope to have an equivalence between IND-Security and RSIM-Security for any functionalities. Nevertheless, we showed an equivalence for PE with public-index.

In this section we go further showing equivalences for PE with *private-index* for several functionalities including Anonymous IBE, inner-product over $\mathbb{Z}_2$, any family of NC_0 circuits, and monotone conjunctive Boolean formulae.

Abstracting the properties needed by the simulator. A closer look at the proof of theorem 4.1 hints some abstract properties that a predicate has to satisfy in order for the simulator to be able to produce an indistinguishable view. We identify the following two properties. The execution of the simulator is divided in *runs*. At run j, the simulator invokes the adversary on input a ciphertext for message x_j, whereas the adversary chose x, and keeps the invariant that x_j gives the same results than x respect to the queries asked by the adversary until that run. At some point the adversary asks a query k for which $F(k,x) \neq F(k,x_j) \neq \perp$ thus the simulator is not able to answer the query in this run. But if the functionality has the property (1) that it is easy to *pre-sample* a new value x_{j+1} that satisfies all queries including the new one, the simulator can rewind the adversary this time on input an encryption of value x_{j+1}. This is still not sufficient since there is no bound on the maximum number of rewinds needed by the simulator so we have to require the property (2) to force the simulation progresses towards a maximum.

To give a clear example, consider how a simulator could work for Anonymous IBE. Suppose that the adversary chooses as challenge identity `crypto` and the simulator chooses `aaaaa` as simulated identity for the ciphertext the simulator will pass to the adversary. Then, the adversary issues a query for identity *bbbbb* and the simulator learns that the predicate is not satisfied against, so the query gives the same evaluation on both the challenge identity and the simulated identity. This is coherent with the query, so the simulator can continue the simulation. Now, suppose that the adversary issues the query for identity `crypto`. Then, the simulated identity is no more compatible with the new query and the simulator has to rewind the adversary but, since the simulator has learnt the challenge identity *crypto* and the corresponding payload exactly, in the next run the simulator is able to finish the simulation perfectly.

In Section 6.2, we show how to implement a more complicated strategy for the predicate inner-product over $\mathbb{Z}_2$.

6.1 Equivalence for Anonymous IBE

The following theorem is an extension of Theorem 4.1.

Theorem 6.1 Let AIBE be an Anonymous IBE scheme (poly, poly, poly)-IND-Secure. Then, AIBE is (poly, poly, poly)-RSIM-Secure as well.

Intuition. Notice that, in an Anonymous IBE scheme the ciphertext does not leak the identity for which it has been generated and thus the special key ϵ does not provide this information as for a public-index scheme. Despite this, when the adversary issues a key-generation query for a key k such that $F(k,x) \neq \perp$, then the simulator *learns* that x is a message for index (or identity

for the case of AIBE) k and payload $F(k,x)$. Thus, the simulator rewinds the adversary on input a freshly generated ciphertext for that pair and can safely generate an *honest* secret key for k upon request.

Another important difference with the proof of Theorem 4.1 is that the simulator could be forced to rewind without gaining any new knowledge and this could result in a never ending simulation. This happens for example in the following case: Let x a challenge message chosen by the adversary and let $x^\star$ the message chosen by the simulator to simulate the ciphertext for x. Then, if the adversary issues a key-generation query for key k such that $F(k,x) = \perp$ but $F(k,x^\star) \neq \perp$, then the simulator is forced to rewind without gaining any new knowledge and this could happen indefinitely. But, the IND-Security of AIBE scheme guarantees that such situation can happen only with negligible probability, and thus the simulator can just halt in such cases.

Proof: Our simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ works as follows. Sim_0 takes in input the master public and secret key, the list $\mathcal{Q} = (k_i, \mathsf{Sk}_{k_i}, F(k_i,\vec{x}))_{i\in[q_1]}$, and the intentionally leaked information about the challenge messages $F(\epsilon,\vec{x}) = (|\mathsf{ind}_j|, |\mathsf{m}_j|)_{j\in[\ell]}$. Then, for each $i \in [q_1]$, Sim_0 checks whatever $F(k_i,x_j) \neq \perp$ for some $j \in [\ell]$. If it is the case, then Sim_0 learns that message x_j is for identity $\mathsf{ind}_j = k_i$ and payload $\mathsf{m}_j = F(k_i,x_j)$.

Let $\mathcal{X}$ the set of tuple of the following form $(j, \mathsf{ind}_j, \mathsf{m}_j)$ learnt by Sim_0. Then, for each pair in $\mathcal{X}$, Sim_0 generates a normal ciphertext for message $x^\star_{\mathsf{j}} = (\mathsf{ind}^\star_j, \mathsf{m}^\star_j)$, with $\mathsf{ind}^\star_j = \mathsf{ind}_j$ and $\mathsf{m}^\star_j = \mathsf{m}_j$, by invoking the encryption algorithm. For all the other positions k for which Sim_0 was not able to learn the corresponding index and payload, Sim_0 generate a ciphertext for random $x^\star_k = (\mathsf{ind}^\star_k, \mathsf{m}^\star_k)$.

Then, Sim_0 executes Adv_1 on input the challenge ciphertexts $(\mathsf{Ct}^\star_j)_{j\in[\ell]}$, where $\mathsf{Ct}^\star_j$ is for message $x^\star_{\mathsf{j}} = (\mathsf{ind}^\star_j, \mathsf{m}^\star_j)$ as described above. When Adv_1 invokes its key-generation oracle on input key k, Sim_1 is asked to generate a corresponding secret key given k and $F(k,\vec{x})$. Now we have the following cases:

1. *If for each $j \in [\ell]$ such that $F(k,x_j) \neq \perp$, $(j,k,F(k,x_j)) \in \mathcal{X}$*: Then we have two sub-cases:
   (a) If there exists and index $\mathsf{j} \in [\ell]$ such that $F(k,x_{\mathsf{j}}) = \perp$ but $F(k,x^\star_{\mathsf{j}}) \neq \perp$ then Sim_0 halts.
   (b) Otherwise, Sim_1 honestly generates a secret key Sk_k for key k. Notice that it holds that $F(k,x^\star_{\mathsf{j}}) = F(k,x_j)$ for all $j \in [\ell]$.
2. *If there exists an index $j \in [\ell]$ such that $F(k,x_j) \neq \perp$ but $(j,k,F(k,x_j)) \notin \mathcal{X}$*: Then $F(k,x^\star_{\mathsf{j}}) \neq F(k,x_j)$ with high probability. Thus Sim_0 adds $(j,k,F(k,x_j))$ to $\mathcal{X}$ and rewinds Adv_1 on freshly generated ciphertexts based on the information Sim_0 has collected in $\mathcal{X}$ so far.
3. *If for all $j \in [\ell]$, $F(k,x_j) = \perp$*: Then we have two sub-cases:
   (a) If there exists and index $\mathsf{j} \in [\ell]$ such that $F(k,x_{\mathsf{j}}) = \perp$ but $F(k,x^\star_{\mathsf{j}}) \neq \perp$ then Sim_0 halts.
   (b) Otherwise, Sim_1 honestly generates a secret key Sk_k for key k. Notice that it holds that $F(k,x^\star_{\mathsf{j}}) = F(k,x_j) = \perp$ for all $j \in [\ell]$.

If after a query the simulator has got to rewind the adversary, we say that such query triggered a rewind. If at some point the adversary halts giving some output, then the simulator outputs what the adversary outputs. This conclude the description of the simulator Sim.

This concludes the description of the simulator.

Let us first bound the probability that the simulator halts during its simulation, this happens in cases 1.(a) or 3.(a). Let us focus on case 1.(a), the other one is symmetric. Notice that when case 1.(a) happens then $F(k, x_{\mathsf{j}}) = \perp$ but $F(k, x^{\star}_{\mathsf{j}}) \neq \perp$, meaning that $\mathsf{ind}_{\mathsf{j}} \neq k$ and $\mathsf{ind}^{\star}_{\mathsf{j}} = k$, and that all the previous key-generation queries are good, meaning that no rewind has been triggered. Therefore, if this event happens with some non-negligible probability, Adv can be used to build another adversary $\mathcal{B}$ that distinguishes between the encryption of x_{j} and $x^{\star}_{\mathsf{j}}$ with the same probability, thus contradicting the IND-Security of the scheme. Precisely, $\mathcal{B}$ simulates the view to $\mathcal{A}$ as described before (i.e., simulating the interface with the simulator) and gives as its challenge two messages with indices $\mathsf{ind}_0 = \mathsf{ind}_j$ and $\mathsf{ind}_1 = \mathsf{ind}^{\star}_j$, where the two indices are as before. Then, $\mathcal{B}$ runs Adv on some ciphertext that is identical to that described before except that $\mathsf{Ct}^{\star}_j$ is set to the challenge ciphertext received from the challenger of the IND-Security game. If at some point Adv asks a query for identity $\mathsf{ind}^{\star}_j$, then $\mathcal{B}$ guesses that the bit 1 was encrypted, otherwise $\mathcal{B}$ outputs 0 as its guess. Notice that if the challenge ciphertext for $\mathcal{B}$ is for the challenge message with identity $\mathsf{ind}_1 = \mathsf{ind}^{\star}_j$, $\mathcal{B}$ perfectly simulated the view of $\mathcal{A}$ when interacting with the above simulator, and thus, by hypothesis on the non-negligible probability of occurence of the case 1.(a), $\mathcal{B}$ outputs 1 with non-negligible probability. On the other hand, if the challenge ciphertext is for the challenge message with identity $\mathsf{ind}_0 = \mathsf{ind}_j$, then the view of Adv is completely independent from $\mathsf{ind}^{\star}_j$, so the probability that Adv asks a query for such identity is negligible and thus $\mathcal{B}$ outputs 0 with overwhelming probability.

Finally, notice that the number of runs, meaning the number of times the simulator rewinds (a rewind happens when case 2. occurs), is upper-bounded by the number of challenge messages ℓ that is polynomial in the security parameter. In fact, each time that a query is not good and the simulator needs to rewind then the simulator learn a new pair $(\mathsf{ind}_j, \mathsf{m}_j)$, for some $j \in [\ell]$ and the same query will never cause a rewind anymore. In the last run, that in which all the key-generation queries are good, the view of the adversary is indistinguishable from that in the real game. This follows from the IND-Security of AIBE by noting that the evaluations of the secret keys on the challenge ciphertexts in the real experiment give the same values than the evaluation of the simulated secret keys on the simulated ciphertexts in the ideal experiment since the secret keys are generated honestly. Therefore, the IND-Security guarantees that in this case the view in the real experiment is indistinguishable from that in the ideal experiment. ▌

6.2 Equivalence for Inner-Product over $\mathbb{Z}_2$ and NC_0

The functionality inner-product over $\mathbb{Z}_2$ (IP, for short)[4] is defined in the following way. It is a family of predicates with key space K_n and index space I_n consisting of binary strings of length n, and for any $k \in K_n, x \in I_n$ the predicate $\mathsf{IP}(k, x) = 1$ if and only if $\sum_{i \in [n]} k_i \cdot x_i = 0 \mod 2$.

Theorem 6.2 If a predicate encryption scheme PE for IP is $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-$\mathsf{IND}$-Secure then PE is $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-$\mathsf{RSIM}$-Secure as well.

Proof: The proof follows the lines of the Theorem 4.1. For simplicity we assume that the adversary outputs a challenge message with the payload set to 1, i.e., the functionality returns values in $\{0, 1\}$, but this can be easily generalized by handling the payload as in the proof of theorem 4.1. Let $x = (x_1, \ldots, x_\ell) \in \{0, 1\}^{n \cdot \ell}$ be the challenge index [5] output by the adversary

[4] We remark that our inner-product is defined over $\mathbb{Z}_2$ so the predicate is different from that of [KSW08].

[5] The challenge index output by the adversary consists of a tuple $(x_1, \ldots, x_\ell)$ of vectors where each element $x_i \in \{0, 1\}^n$ for $i = 1, \ldots, \ell$. For simplicity, henceforth we interpret such challenges as vectors in $\{0, 1\}^{n \cdot \ell}$.

Adv_0 and let $(w_i)_{i=1}^{q_1}$ be the queries asked by Adv_0 (i.e. the queries asked before seeing the challenge ciphertexts). As usual we divide the execution of the simulator in runs and in any run the simulator keeps an index $x^0 = (x_1^0, \ldots, x_\ell^0) \in \{0,1\}^{n \cdot \ell}$ that uses to encrypt the simulated ciphertext given to the adversary in that run. Let Y_i be a matrix in $\{0,1\}^{(q_1+i-i)\times n}$ where the rows $y_1, \ldots, y_{q_1+i-1}$ of Y_i are such that the first q_1 rows $y_1, \ldots, y_{q_1}$ consist of the vectors $w_1, \ldots, w_{q_1}$ (i.e., $y_1 = w_1, \ldots, y_{q_1} = w_{q_1}$) and for each $j = 1, \ldots, i-1$ the row y_{q_1+j} of Y_i corresponds to the last query asked by Adv_1 in run j (as it will be clear soon, in any run i, if the last query asked by the adversary in such run will trigger a rewind, then only such query is put in the matrix, and not any other previous query asked by the adversary in run i). Furthermore, for any $i \geq 1$ and any $j \in [\ell]$, let $b_{i,j} \in \{0,1\}^{q_1+i-1}$ be the column vector such that $b_{i,j}[k] = \mathsf{IP}(y_k, x_j), k = 1, \ldots, q_1 + i - 1$. During the course of the simulation, the simulator will guarantee the following invariant: at the beginning of any run $i \geq 1$, for any $j \in [\ell]$, $Y_i \cdot x_j^0 = b_{i,j}$. In the first run the simulator runs the adversary with input a ciphertext that encrypts an index $x^0 = (x_1^0, \ldots, x_\ell^0) \in \{0,1\}^{n \cdot \ell}$ such that for any $j \in [\ell]$, $Y_1 \cdot x_j^0 = b_{1,j}$. The simulator can efficiently find such vector by using the PS of IP guaranteed by Theorem E.2. When in a run $i \geq 1$ the adversary makes a query for a vector $y \in \{0,1\}^n$ we distinguish two mutually exclusive cases. executed).

1. *The vector y is a linear combination of the rows of Y_i.* Then, by the invariant property it follows that for any $j \in [\ell]$, $\mathsf{IP}(y, x_j) = \mathsf{IP}(y, x_j^0)$, and the simulator continues the simulation answering the query as usual (i.e., by giving to the adversary Adv_1 the secret key for y generated honestly).

2. *The vector y is not a linear combination of the rows of Y_i.* Then, the simulator could not be able to answer this query. In this case, we say that the query triggered a rewind and the simulator rewinds the adversary Adv_1 as follows. The simulator updates Y_{i+1} by adding the new row y to Y_i and uses the PS of IP guaranteed by theorem E.2 to efficiently find a new vector $x' = (x'_1, \ldots, x'_\ell) \in \{0,1\}^{n \cdot \ell}$ such that for any $j \in [\ell]$, $Y_{i+1} \cdot x'_j = b_{i+1,j}$ (i.e., the PS algorithm is invoked independently for each equation $Y_{i+1} \cdot x'_j = b_{i+1,j}$). Finally, the simulator rewinds the adversary by invoking it with input the encryption of x' and updates x^0 setting it to x'. Notice that at the beginning of run $i+1$ the invariant is still satisfied.

At the end of the last run, the simulator outputs what the adversary outputs. It is easy to see that the simulator executes at most n runs since in any run $i > 2$ the rank Y_i is greater than the rank of Y_{i-1} and for any $i \geq 1$ the rank of Y_i is at most n. Finally, notice that at the beginning of the last run the invariant guarantees that for any query y asked by Adv_0 and for any $j \in [\ell]$ $\mathsf{IP}(y, x_j) = \mathsf{IP}(y, x_j^0)$. Furthermore, since in the last run no query has triggered a rewind, then any query asked by Adv_1 in the last run still satisfies this property. Therefore, by the IND-Security of the scheme, it follows that the output of the simulator is indistinguishable from that of the adversary in the real game. ∎

Recall that NC_0 is the class of all family of Boolean circuits of polynomial size and constant depth with AND, OR, and NOT gates of fan-in at most 2. It is a known fact that circuits in NC_0 with n-bits input and one-bit output can be expressed as multivariate polynomials $p(x_1, \ldots, x_n)$ over $\mathbb{Z}_2$ of constant degree. Furthermore, you can encode such polynomials as vectors in $\mathbb{Z}_2^{n^m}$ for some constant m and evaluate them at any point using the inner-product predicate. Therefore, it is easy to see that the previous proof extends naturally to family of circuits in NC_0 but we omit the details.

Theorem 6.3 If a predicate encryption scheme PE for a family of circuits in NC_0 wit one-bit output is (poly, poly, poly)-IND-Secure then PE is (poly, poly, poly)-RSIM-Secure as well.

6.3 Equivalence for Monotone Conjunctive Boolean Formulae

The functionality Monotone Conjunctive Boolean Formulae (MCF, for short) is defined in the following way. It is a family of predicates with key space K_n consisting of monotone (i.e., without negated variables) conjunctive Boolean formulae over n variables (i.e., a subset of indices in $[n]$) and index space I_n consisting of assignments to n Boolean variables (i.e., binary strings of length n), and for any $\phi \in K_n, x \in I_n$ the predicate $\mathsf{MCF}(\phi, x) = 1$ if and only if the assigment x satisfies the formula ϕ. If a formula $\phi \subseteq [n]$ contains the index i, we say that ϕ has the i-th formal variable set.

The reader may have noticed that PE for MCF is a special case of PE for the family of all conjunctive Boolean formulae introduced by [BW07]. Though the monotonicity weakens the power of the primitive, it has still interesting applications like PE for subset queries as shown by [BW07]. We point out that the monotonicity is fundamental to implement our rewinding strategy. In fact, (under some complexity assumption) the functionality that computes the family of all conjunctive Boolean formulae is not PS[6], so it is not clear whether an equivalence between (poly, poly, poly)-IND-Security and (poly, poly, poly)-RSIM-Security can be established for this primitive. On the other hand, weakening the functionality to only allow monotone formulae we are able to prove the following theorem.

Theorem 6.4 If a predicate encryption scheme PE for MCF is (poly, poly, poly)-IND-Secure then PE is (poly, poly, poly)-RSIM-Secure as well.

Proof Sketch. The proof follows the lines of the previous equivalence theorems and is only sketched outlining the differences. Let $x = (x_1, \ldots, x_\ell)$ be the challenge index (i.e., assignment) vector chosen by the adversary Adv_0 that the simulator does not know. The simulator can easily sample an index vector $x^0 = (x_1^0, \ldots, x_\ell^0)$ such that for any $i \in [\ell]$, x_i^0 satisfies the equations: $\mathsf{MCF}(\phi, x_i^0) = \mathsf{MCF}(\phi, x_i)$ for any query ϕ asked by Adv_0 before seeing the challenge ciphertexts. This can be done by the simulator in the following way just having the evaluations of the assignments on the formulae. In full generality, fix an arbitrary set of formulae $A = \{\phi_i\}_{i \in [q]}$ and their evaluations over some (hidden) assignment $x = (x_1, \ldots, x_\ell)$. For any $j \in [\ell]$ and any position $k \in [n]$, the simulator sets the k-th bit of x_j^0 to be 1 or 0 according to the following rules. If there exists some $\phi \in A$ that has the k-th formal variable set *and* x_j satisfies ϕ (the simulator has this information because it knows the evaluation of ϕ on x_j), then the k-th bit of x_j^0 is set to 1, otherwise (i.e., whether either the k-th formal variable of ϕ is not set or x_j does not satisfy ϕ) it is set to 0. It is easy to see that x^0 satisfies the previous equations with respect to the set of formulae A and thus is a valid pre-image of x. As usual, we divide the execution of the simulation in runs. During the course of the simulation, the simulator will guarantee the invariant that at the beginning of any run, the index vector x^0 satisfies all equations with respect to the (hidden) vector x and to all queries asked by the adversary. If a new query does not satisfy such equations, then the simulator has to find a new pre-image that satisfies all the equations including the new one. This is done as before by pre-sampling according to the above rules. Notice that once a bit in some index x_j^0 is set to 1, it is not longer changed. Thus, it follows that the number of runs is upper bounded by the bit length of x. Therefore, if PE is IND-Secure, the simulator can conclude the simulation and produce an output indistinguishable from that of the adversary as desired. □

[6]The authors of [DIJ+13] proved this fact that will appear in the full version of their paper.

6.4 Predicates with Polynomial Size Key Space

Boneh *et al.* [BSW11] (see also [BW07]) presented a generic construction for functional encryption for any functionality F where the key space K has polynomial size that can be proven $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-IND-Secure in the standard model and a modification that can be proven $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-SIM-Secure in the random oracle model. Bellare and O'Neill [BO12] proved the $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-SIM-Security of their scheme assuming that the underlying PKE scheme is secure against key-revealing selective opening attack (SOA-K) [BDWY12]. On the other hand we prove that the construction is $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-RSIM-Secure assuming only IND-CPA PKE that is a weaker assumption than SOA-K PKE needed in [BO12].

The construction of Boneh *et al.* is the following. Let $s = |K| - 1$ and $K = (k_0 = \epsilon, k_1, \ldots, k_s)$.[7] The brute force functional encryption scheme realizing F uses a semantically secure public-key encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and works as follows:

1. $\mathsf{Setup}(1^\lambda)$: for $i = 1, \ldots, s$, run $(\mathcal{E}.\mathsf{pk}_i, \mathcal{E}.\mathsf{sk}_i) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and output $\mathsf{Mpk} = (\mathcal{E}.\mathsf{pk}_1, \ldots, \mathcal{E}.\mathsf{pk}_s)$ and $\mathsf{Msk} = (\mathcal{E}.\mathsf{sk}_1, \ldots, \mathcal{E}.\mathsf{sk}_s)$.

2. $\mathsf{KeyGen}(\mathsf{Msk}, k_i)$: output $\mathsf{sk}_i := \mathcal{E}.\mathsf{sk}_i$.

3. $\mathsf{Enc}(\mathsf{Msk}, x)$: output $\mathsf{Ct} := (F(\epsilon, x), \mathcal{E}.\mathsf{Enc}(\mathcal{E}.\mathsf{pk}_1, F(k_1, x)), \ldots, \mathcal{E}.\mathsf{Enc}(\mathcal{E}.\mathsf{pk}_s, F(k_s, x)))$.

4. $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{Ct})$: output $\mathsf{Ct}[0]$ if $\mathsf{sk}_i = \epsilon$, and output $\mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{sk}_i, \mathsf{Ct}[i])$ otherwise.

Theorem 6.5 Let FE be the above $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-IND-Secure functional encryption scheme for the functionality F. Then, FE is $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-RSIM-Secure as well.

Proof Sketch. The proof uses the same ideas of those in the Sections 4 and 6. Roughly, the strategy of the simulator is the following. Again, we divide the execution of the simulator in runs. Let $(x_1, \ldots, x_\ell)$ be the vector of challenge messages chosen by the adversary and unknown to the simulator. At the beginning of the first run, the simulator executes the adversary on input ciphertexts $(\mathsf{Ct}_1, \ldots, \mathsf{Ct}_\ell)$ that encrypt dummy values. Recall that for any $i \in [\ell]$, $\mathsf{Ct}_i[j]$ is supposed to encrypt $F(k_j, x_i)$. When the adversary issue a key-generation query k_j, the simulator learns $(F(k_j, x_1), \ldots, F(k_j, x_\ell))$. Then, the simulator rewinds the adversary executing it with input a new tuple of ciphertexts $(\mathsf{Ct}'_1, \ldots, \mathsf{Ct}'_n)$ where for each $i \in [\ell], j = 1, \ldots, s$, $\mathsf{Ct}'_i[j]$ encrypts $F(k_j, x_i)$. After at most $s + 1$ runs, the simulated ciphertext encrypts the same values as in the real game, and the simulator terminates returning the output of the adversary. This concludes the proof. □

FE with multi-bit output. Notice that a predicate encryption scheme for predicate P implies a predicate encryption scheme for the same predicate where the payload is fixed to 1 (meaning that the predicate is satisfied). This in turn implies a functional encryption for the functionality P (where the evaluation algorithm of the FE scheme runs the evaluation algorithm of the PE scheme and outputs 0 if the PE scheme returns $\perp$ and 1 otherwise). Finally, the latter implies a functional encryption scheme for the class of circuits with multi-bit output that extends P in the obvious way. These implications preserve the $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-RSIM- Security.

[7] For sake of simplicity we implicitly assume that the functionality is not parametrized by the security parameter but this can be generalized easily.

7 Impossibility of RSIM for FE for General Circuits

In this section, we show an impossibility result for RSIM-Security in the *auxiliary input setting* in the *standard model* adapting the impossibility results of [BSW11, BO12, AGVW13]. Specifically, the main theorem of this section is the following:

Theorem 7.1 Assuming the existence of collision resistance hash functions and pseudo-random functions, there exists a family of circuits for which there are no $(0, 1, \mathsf{poly})$-RSIM-Secure functional encryption schemes in the auxiliary input setting in the standard model.

Overview. The main difference with previous impossibility results is that for RSIM the trade-off between the length of the secret keys and the total number of bits in messages securely encrypted shows up only when considering richer class of functionality containing at least pseudo-random functions. In this sense, our results can be seen as the *dual* of that of [AGVW13]. Specifically, we consider an adversary that issues a suitable number of challenge messages, says $kl(\lambda) + \lambda$, where $kl(\cdot)$ is the polynomial bounding the length of the secret keys, of the type $(s||r_i)_{i\in[\ell]}$ where s will be the seed of the pseudo-random function and r_i a random value that will be part of the input on which the pseudo-random function will be evaluated. Then the adversary, on input Mpk and the ciphertexts $(\mathsf{Ct}_i)_{i\in[\ell]}$ for the challenge messages, issues a single adaptive key-generation query to its oracle for the circuit $C^{\mathsf{PRF},w}$ that computes the pseudo-random function on input seed s and value $r||w$, where $w = \mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1||\cdots||\mathsf{Ct}_\ell)$ is hardwired in $C^{\mathsf{PRF},w}$ and is used to commit the simulator to the ciphertexts it has generated. Crucial is the fact that the output of $C^{\mathsf{PRF},w}$ on the challenge messages depends on the Ct_i's, this will defeat the power of rewinding given to the simulator.

Proof: Le FE be a $(0, 1, \mathsf{poly})$-RSIM-Secure functional encryption scheme for circuits with secret-key length $kl(\cdot)$. Let $\mathsf{PRF} = \{\mathsf{PRF}_\lambda : \{0,1\}^\lambda \times \{0,1\}^{2\cdot m(\lambda)} \to \{0,1\}\}_{\lambda\in\mathbb{N}}$ a circuit family of pseudo-random functions. Let CRHF be the collision resistance hash function with range $m(\lambda)$ whose key hk has been chosen by the auxiliary input generator. We omit hk in the notation just for the sake of simplicity.

Consider the following adversary $\mathsf{Adv} = (\mathsf{Adv}_0, \mathsf{Adv}_1)$ and distinguisher $\mathcal{D}$ in the $(0, 1, \mathsf{poly})$-RSIM security experiment. Specifically, Adv works as follows:

- Adv_0 returns $\ell = kl(\lambda) + \lambda$ challenge messages of the form $(s||r_i)$ for random $s \in \{0,1\}^\lambda$ and $r_i \in \{0,1\}^{m(\lambda)}$.
- Adv_1, on input Mpk, $(\mathsf{Ct}_i)_{i\in[\ell]}$ and $\mathtt{st}$, sets $w = \mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1||\cdots||\mathsf{Ct}_\ell)$ and invokes the key-generation oracle on input the circuit $C^{\mathsf{PRF},w}(s, r) := \mathsf{PRF}(s, r||w)$, and obtains secret key Sk for it. Finally, Adv_1 outputs $\alpha = ((\mathsf{Ct}_i)_{i\in[\ell]}, w, \mathsf{Sk})$.

Instead, the distinguisher $\mathcal{D}$ does the following:

- $\mathcal{D}$, on input Mpk, the challenge messages $(s||r_i)_{i\in[\ell]}$ and α, interprets α as $\alpha = ((\mathsf{Ct}_i)_{i\in[\ell]}, w,$ Sk) and checks that (1) w is equal to $\mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1||\cdots||\mathsf{Ct}_\ell)$, and (2) $\mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}_i, \mathsf{Sk}) = \mathsf{PRF}(s, r_i||w)$ for each $i \in [\ell]$. $\mathcal{D}$ returns 1 if all the checks passed, 0 otherwise.

Because we assumed FE to be $(0, 1, \mathsf{poly})$-RSIM-Secure, it means there exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ that generates a view indistinguishable to that of Adv when it plays in the real game. Given this simulator, we now construct an adversary $\mathcal{A}$ against the security of the pseudo-random function. Specifically, $\mathcal{A}$ on input the security parameter 1^λ and given access to oracle $\mathcal{O}$ does the following:

$\mathcal{A}^{\mathcal{O}}(1^\lambda)$:

1. $\mathcal{A}$ invokes the setup algorithm of FE to generate master public and secret key. Namely, $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. Let $\ell = kl(\lambda) + \lambda$. Then $\mathcal{A}$ chooses random $r_i \in \{0,1\}^{m(\lambda)}$ for $i \in [\ell]$, as Adv_0 does.
3. $\mathcal{A}$ runs Sim_0 on input $(\mathsf{Mpk}, \mathsf{Msk}, \mathcal{Q}, (F(\epsilon, (s||r_i)))_{i\in[\ell]})$, where $\mathcal{Q}$ is empty because Adv_0 does not issue any key-generation query. When Adv_1 invokes its key-generation oracle on input circuit $C^{\mathsf{PRF},w}$, $\mathcal{A}$ invokes Sim_1 on input $C^{\mathsf{PRF},w}$ and $(\mathcal{O}(r_i||w))_{i\in[\ell]}$ as input.

   At some point Sim_0 returns α.
4. Finally, $\mathcal{A}$ does the same checks as $\mathcal{D}$. Namely, $\mathcal{A}$ interprets α as $\alpha = ((\mathsf{Ct}_i)_{i\in[\ell]}, w, \mathsf{Sk})$ and checks that
   (a) w is equal to $\mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1||\cdots||\mathsf{Ct}_\ell)$, and
   (b) $\mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}_i, \mathsf{Sk}) = \mathcal{O}(r_i||w)$ for each $i \in [\ell]$.

   $\mathcal{A}$ returns 1 if all the checks passed, 0 otherwise.

Now observe that, $\mathcal{D}$ outputs 1 with overwhelming probability when given the output of adversary Adv in the $(0,1,\mathsf{poly})$-RSIM real experiment. Moreover, by the $(0,1,\mathsf{poly})$-RSIM-Security of FE, $\mathcal{D}$ also output 1 with overwhelming probability when given the output of the simulator Sim. Then, if $\mathcal{O}$ is the pseudo-random oracle for random seed s, $\mathcal{A}$ perfectly simulates the output of Sim in the $(0,1,\mathsf{poly})$-RSIM ideal experiment and thus $\mathcal{A}$ gives in output 1 with high probability.

Suppose now that $\mathcal{O}$ is a truly random oracle. Let $\alpha = ((\mathsf{Ct}_i)_{i\in[\ell]}, w, \mathsf{Sk})$ be the output of Sim during the execution of $\mathcal{A}$ (see point (3) in the description of $\mathcal{A}$). We distinguish two mutually exclusive cases.

1. Adv_1 *has never ever issued a key-generation query for circuit* $C^{\mathsf{PRF},w}$. In this case the probability that $\mathcal{A}$ outputs 1 is negligible since the output of the simulator is independent from $\mathcal{O}(r_i||w)$ for each $i \in [\ell]$ and these values are random being $\mathcal{O}$ a truly random oracle.
2. Adv_1 *invoked its key-generation oracle on input the circuit* $C^{\mathsf{PRF},w}$ *at least one time.* First, notice that $\mathcal{A}$ implements the interface between Adv_1 and Sim. Precisely, when Sim_0 invokes its oracle on some input, then $\mathcal{A}$ invokes Adv_1 on the same input. Then, when Adv_1 issues a key-generation query for a circuit $C^{\mathsf{PRF},w}$, $\mathcal{A}$ sees the value w and answers such query as described above.
   Let $p(\lambda)$ be the running time of Sim. Therefore, the execution of Sim can be divided in at most $p(\lambda)$ *runs*, where for $j = 1, \ldots, p(\lambda)$, in the j-th run Sim_0 invokes its oracle on input $(\mathsf{Ct}_i^j)_{i\in[\ell]}$ that corresponds to a key-generation query for circuit C^{PRF,w_j}, where $w_j = \mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1^j||\ldots||\mathsf{Ct}_\ell^j)$. Now notice that there exists some index $k \leq p(\lambda)$ such that $w = w_k$ and k is the first index for which $w = w_k$. From this fact and from the fact that $\mathcal{A}$ checks whether $w = \mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1||\ldots||\mathsf{Ct}_\ell)$, it follows that with all but negligible probability $(\mathsf{Ct}_i) = (\mathsf{Ct}_i^k)$. Indeed, suppose towards a contradiction that with non-negligible probability q it holds that $(\mathsf{Ct}_i) \neq (\mathsf{Ct}_i^k)$. Then, Adv_1 and Sim can be used to build an adversary $\mathcal{B}$ for CRHF as follows. $\mathcal{B}$ on input the security parameter 1^λ and

the hash key hk does the following:

$\mathcal{B}(\mathsf{hk})$:

(a) $\mathcal{B}$ invokes the setup algorithm of FE to generate master public and secret key, namely $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, Then, $\mathcal{B}$ initializes a list L to empty and set a global index j to zero. The list L is used by $\mathcal{B}$ to trace the invocations to Adv_1 made by Sim_0.

(b) Let $\ell = kl(\lambda) + \lambda$. Then $\mathcal{B}$ chooses random $r_i \in \{0,1\}^{m(\lambda)}$ for $i \in [\ell]$, as Adv_0 does.

(c) $\mathcal{B}$ runs Sim_0 on input $(\mathsf{Mpk}, \mathsf{Msk}, \mathcal{Q}, (F(\epsilon, (s||r_i)))_{i\in[\ell]})$, where $\mathcal{Q}$ is empty because Adv_0 does not issue any key-generation query. When Adv_1 is invoked on input ciphertexts $(\mathsf{Ct}_i^j)_{i\in[\ell]}$ then $\mathcal{B}$ put an entry in the list L corresponding to

$$\left((\mathsf{Ct}_i^j)_{i\in[\ell]}, w_j = \mathsf{CRHF}(\mathsf{Mpk}||\mathsf{Ct}_1^j||\ldots||\mathsf{Ct}_\ell^j)\right) ,$$

and increment the global index j by one. Then, when Adv_1 invokes its key-generation oracle on input circuit C^{PRF,w_j}, $\mathcal{B}$ invokes Sim_1 on input C^{PRF,w_j} and $(\mathsf{PRF}(s, r_i||w_j))_{i\in[\ell]}$ as input.

At some point Sim_0 returns α.

(d) At this point, $\mathcal{B}$ interprets α as $\alpha = ((\mathsf{Ct}_i)_{i\in[\ell]}, w, \mathsf{Sk})$ and looks up in the list L for the first index k such that $w_k = w$. If $\mathcal{B}$ does not find this index it aborts, otherwise $\mathcal{B}$ returns the pair $((\mathsf{Mpk}||\mathsf{Ct}_1^k||\ldots||\mathsf{Ct}_\ell^k),\ (\mathsf{Mpk}||\mathsf{Ct}_1||\ldots||\mathsf{Ct}_\ell))$ as its collision.

It is easy to see that the probability that $\mathcal{B}$ finds a collision is exactly q.
Finally, notice that, when Sim_0 invokes Adv_0, its view is independent from the values $\mathcal{O}(r_i||w)$'s. This is because, being $\mathcal{O}$ a truly random oracle, for any $j < k$, $w_j \neq w_k = w$ and thus the values $\mathcal{O}(r_i||w_j)$'s are randomly and independently chosen from the values $\mathcal{O}(r_i||w)$'s. Thus, the tuple of ciphertexts $(\mathsf{Ct}_i)_{i\in[\ell]}$ is independent from the tuple $(\mathcal{O}(r_i||w))_{i\in[\ell]}$: we call this Fact 1.
We now bound the probability of the following event E which is defined to be the event that for any $i \in [\ell]$, $\mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}_i, \mathsf{Sk}) = \mathcal{O}(r_i||w)$, where the probability is taken over the random choices of $\mathcal{A}$ (and thus of Adv_1 and Sim) and of the oracle $\mathcal{O}$.

$$\begin{aligned}
\Pr[\,\mathsf{E}\,] &\leq \Pr[\,\exists\ \mathsf{Sk} : |\mathsf{Sk}| = kl(\lambda) \text{ and } \forall i \in [\ell]\ \mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}_i, \mathsf{Sk}) = \mathcal{O}(r_i||w)\,] \\
&\leq \sum_{\mathsf{Sk}\in\{0,1\}^{kl(\lambda)}} \Pr[\,\forall i \in [\ell]\ \mathsf{Eval}(\mathsf{Mpk}, \mathsf{Ct}_i, \mathsf{Sk}) = \mathcal{O}(r_i||w)\,] \text{ (by the union bound)} \\
&\leq \sum_{\mathsf{Sk}\in\{0,1\}^{kl(\lambda)}} 2^{-\ell} \text{ (since Fact 1 holds and } \mathcal{O} \text{ is a truly random oracle)} \\
&\leq 2^{kl(\lambda)-\ell} = 2^{-\lambda} \text{ (since } \ell = kl(\lambda) + \lambda).
\end{aligned}$$

Then, it follows that when $\mathcal{O}$ is a truly random oracle, the probability that $\mathcal{A}^{\mathcal{O}}$ outputs 1 is negligible in the security parameter. Therefore, $\mathcal{A}^{\mathcal{O}}$ can tell apart a pseudorandom oracle from a truly random oracle with non-negligible probability. This concludes the proof. ∎

8 Open problems and Future Work

We leave to future work a full (non-trivial) characterization of the class of functionalities for which the equivalence between IND-Security and RSIM-Security holds, and more in general, of the functionalities F for which there exists a FE scheme for F that is RSIM-Secure. Notice that we provided an equivalence between the two notions for the functionality that computes all circuits in NC_0, and for functionalities like IBE that are computed by a family of circuits in AC_0 (that includes NC_0), but at the same time we also showed an impossibility result for any functionality that computes a pseudorandom function. Naor and Reingold [**?**] constructed pseudorandom functions in TC_0 (that includes AC_0). On the other hand, in a previous work, Linial, Mansour and Nisan [**?**] proved that there do not exist pseudorandom functions in AC_0. So, it is an open question whether there exists (under some reasonable complexity assumption) a $(0, 1, \mathsf{poly})$-RSIM-Secure FE scheme for all circuits in AC_0. Another research direction not explored in our work is to study the power of rewinding simulators in the context of the new recent concepts of randomized functionalities [**?**] and multi-input functionalities [**?**].

Acknowledgments

Vincenzo thanks Sadeq Dousti for suggesting him this line of research and for helpful discussions.

References

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. Cryptology ePrint Archive, 2005. `http://eprint.iacr.org/`.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/`.

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497, Zurich, Switzerland, February 9–11, 2010. Springer, Berlin, Germany.

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

[BCP13] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. Cryptology ePrint Archive, Report 2013/650, 2013. `http://eprint.iacr.org/`.

[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.

[BDWY12] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 645–662, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany.

[BO12] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. Cryptology ePrint Archive, Report 2012/515, 2012. `http://eprint.iacr.org/`.

[BR13] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. *IACR Cryptology ePrint Archive*, 2013:563, 2013.

[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.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer, Berlin, Germany.

[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 *CRYPTO (2)*, pages 519–535, 2013.

[GGH+13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *IACR Cryptology ePrint Archive*, 2013. (To appear in FOCS 2013).

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 479–499. Springer, 2013.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304, 1985.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 89–98, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309.

[GVW12] 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.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554, 2013.

[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.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer, Berlin, Germany.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of

Lecture Notes in Computer Science, pages 457–473, Aarhus, Denmark, May 22–26, 2005. 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.

A Standard Notions

A.1 Pseudo-random function family

Definition A.1 [Pseudo-random function family] A family $\mathsf{PRF} = \{\mathsf{PRF}_s : s \in \{0,1\}^\lambda\}_{\lambda\in\mathbb{N}}$ is called family of $(l(\lambda), L(\lambda))$-*pseudo- random function family* if:

- *Efficiently computable*: For any $\lambda \in \mathbb{N}$, $s \in \{0,1\}^\lambda$, $\mathsf{PRF}_s : \{0,1\}^{l(\lambda)} \to \{0,1\}^{L(\lambda)}$ is polynomial time computable.
- *Pseudo-random*: For any p.p.t adversary $\mathcal{A}$, it holds that:

$$\left|\text{Prob}[\mathcal{A}^{\mathsf{PRF}_s}(1^\lambda) = 1 | s \leftarrow \{0,1\}^\lambda] - \text{Prob}[\mathcal{A}^F(1^\lambda) = 1 | F \leftarrow \mathcal{R}(l(\lambda), L(\lambda))]\right| \leq \mathsf{neg}(\lambda) \ ,$$

  where $\mathcal{R}(l(\lambda), L(\lambda))$ is the space of all possible functions $F : \{0,1\}^{l(\lambda)} \to \{0,1\}^{L(\lambda)}$.

A.2 Collision-resistant hash functions

Definition A.2 A collision-resistant hash function family $\mathsf{CRHF} = \{\mathsf{CRHF}_\lambda : \{0,1\}^\lambda \times D_\lambda \to R_\lambda\}_{\lambda\in\mathbb{N}}$ for $|R_\lambda| < |D_\lambda|$ is a collection of functions satisfying:

- There is a PPT algorithm K that on input 1^λ outputs a random key $\mathsf{hk} \in \{0,1\}^\lambda$.
- There is a deterministic polynomial time algorithm H that for any λ on input a key $\mathsf{hk} \in \{0,1\}^\lambda$ and $x \in D_\lambda$ outputs $\mathsf{CRHF}(\mathsf{hk}, x) = \mathsf{CRHF}^\lambda(\mathsf{hk}, x)$.
- For any PPT algorithm $\mathcal{A}$,

$$\Pr[\mathsf{CRHF}(\mathsf{hk}, x_1) = \mathsf{CRHF}(\mathsf{hk}, x_2) \text{ and } x_1 \neq x_2 | \mathsf{hk} \leftarrow K(1^\lambda); x_1 \leftarrow D_\lambda; x_2 \leftarrow A(\mathsf{hk}, x_1)] \ ,$$

  is negligible in λ.

B Known Simulation-Based Definitions

Notation. $A^{B(\cdot)[[x]]}$ means that the algorithm A can issue a query q to its oracle, at which point $B(q, x)$ will be executed and output a pair (y, x'). The value y is then communicated to A as the response to its query, and the variable x is set to x', and this updated value is fed to the algorithm B the next time it is queried as an oracle, and fed to any algorithms executed later in an experiment that want x as an input.
Also, $A^{B^\circ(\cdot)}$ means that A can send a query q to its oracle, at which point $B^\circ(q)$ is executed, and any oracle queries that B makes are answered by A.

Definition B.1 [Boneh *et al.* [BSW11] Simulation-Based Definition] A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, X) is *simulation-secure* if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for all PPT *adversary* algorithms $\mathsf{Adv} = (\mathsf{Adv}_0, \mathsf{Adv}_1)$ the outputs of the following two experiments are computationally indistinguishable.

$\mathsf{RealExp}^{\mathsf{FE},\mathsf{Adv}}(1^\lambda)$	$\mathsf{IdealExp}^{\mathsf{FE},\mathsf{Adv}}_{\mathsf{Sim}}(1^\lambda, 1^n)$
$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda);$	$(\mathsf{Mpk}, \sigma) \leftarrow \mathsf{Sim}_0(1^\lambda);$
$(\vec{x}, \mathtt{st}) \leftarrow \mathsf{Adv}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk});$	$(\vec{x}, \mathtt{st}) \leftarrow \mathsf{Adv}_0^{\mathsf{Sim}_1(\cdot)[[\sigma]]}(\mathsf{Mpk});$
$\vec{\mathsf{Ct}} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, \vec{x});$	
$\alpha \leftarrow \mathsf{Adv}_1^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk}, \vec{\mathsf{Ct}}, \mathtt{st});$	$\alpha \leftarrow \mathsf{Sim}_2^{F(\cdot,\vec{x}),\mathsf{Adv}_1^{\circ}(\mathsf{Mpk},\cdot,\mathtt{st})}(\sigma, F(\epsilon, \vec{x}));$
Let $y_1, \ldots, y_q$ the oracle queries made by Adv	Let $y_1, \ldots, y_q$ the oracle queries to F made by Sim_2
Output: $(\mathsf{Mpk}, \vec{x}, \mathtt{st}, \alpha, y_1, \ldots, y_q)$	**Output:** $(\mathsf{Mpk}, \vec{x}, \mathtt{st}, \alpha, y_1, \ldots, y_q)$

Definition B.2 [De Caro *et al.* [DIJ+13] Simulation-Based Definition] A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, M) is *simulation-secure* if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for all *adversary* algorithms $\mathsf{Adv} = (\mathsf{Adv}_0, \mathsf{Adv}_1)$ the outputs of the following two experiments are computationally indistinguishable.

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

Here, the (k_i)'s correspond to the key-generation 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 secret key, 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.

C Positive result for Unbounded Simulation

In this section we prove a positive result for simulation-based security with respect to unbounded simulators (USIM-Security). We point out that a similar fact was noticed by [BR13] in the context of obfuscation. Formally, we have the following definition.

Definition C.1 We say that a FE scheme is USIM-Secure if it is secure according to definition B.2 against PPT[8] adversaries and with respect to simulators of unbounded computational power.

Theorem C.2 If a FE scheme FE for functionality $F : K \times X \to \Sigma$ is $(\mathsf{poly}, 0, \mathsf{poly})$-IND-Secure then it is $(\mathsf{poly}, 0, \mathsf{poly})$-USIM-Secure.

Proof: The proof is a generalization of the result [O'N10]. Suppose that there exists an adversary $\mathcal{A}$ that breaks the SIM-Security of definition B.2. We build a simulator Sim of unbounded computational power that will be used to break the IND-Security of FE. The simulator Sim takes as input the public-key and runs the adversary with it. Recall that in our definition, when the adversary makes a query k to the oracle, it is answered by the oracle *honestly* (and not generated by the simulator). Therefore, after the query phase, the view of $\mathcal{A}$ in the real experiment is identical to that in the simulated experiment. Let $k_1, \ldots, k_q$ the queries requested by the adversary to the oracle. At some point, $\mathcal{A}$ submits its challenge vector $x = (x_1, \ldots, x_m)$ that the simulator does not see. Anyway, the simulator Sim receives as input the public-key and the set of pairs $(k_i, y_{i,j} = F(k_i, x_j))$ for any $j \in [m]$ and any $i \in [q]$, where q is the number of queries asked by the adversary Adv_0. Sim uses its unbounded computational power to search over all message space X for a (possibly different) message vector $x' = (x'_1, \ldots, x'_m)$ that verifies the equations $y_i = F(k_i, x'_j)$ for any $j \in [m]$ and any $i \in [q]$. Notice that at least one such message vector exists, namely message $x \in X$, thus the simulator can find it. The simulator Sim outputs as ciphertext an encryption of x' generated with the public-key. It is easy to see that if FE is IND-Secure, then the adversary cannot distinguish the view generated by the simulator from a view in the real experiment. The proof of this fact is identical to that given in [O'N10] for the equivalence of IND-Security and non-adaptive SIM-Security for pre-image sampleable functionalities and the details can be found there. This concludes the proof. ∎

D RSIM-Security $\implies$ IND-Security

Theorem D.1 Let FE be a functional encryption scheme that is RSIM-Secure, then FE is IND-Secure as well.

Proof: Suppose towards a contradiction that there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that breaks the IND-Security of FE. Consider the following adversary $\mathcal{B}^b = (\mathcal{B}_0^b, \mathcal{B}_1^b)$, for $b \in \{0, 1\}$, and distinguisher $\mathcal{D}$, for the RSIM-Security of FE.

[8] We stress that a definition with unbounded adversaries is impossible to achieve. In fact, it is possible to prove that if the FE scheme is correct and provides a key that implements the identity function, there is an adversary that can recover the whole message from a ciphertext.

- $\mathcal{B}_0^b$, on input master public key Mpk and having oracle access to the key-generation oracle, invokes $\mathcal{A}_0$ on input Mpk and emulates $\mathcal{A}_0$'s key-generation oracle by using its own oracle.

  When $\mathcal{A}_0$ stops, it outputs two *challenge messages vectors*, of length ℓ, $\vec{x}_0, \vec{x}_1 \in X^\ell$ and its internal state $\mathtt{st}$.

  hen, $\mathcal{B}_0^b$ outputs $(\vec{x}_b, \mathtt{st}' = (\mathtt{st}, b, \vec{x}_{1-b}))$.

- $\mathcal{B}_1^b$, on input master public key Mpk, challenge ciphertexts $(\mathsf{Ct}_i)_{i\in[\ell]}$ and state $\mathtt{st}' = (\mathtt{st}, b, \vec{x}_{1-b})$ and having oracle access to the key-generation oracle, invokes $\mathcal{A}_1$ on input the challenge ciphertexts and state $\mathtt{st}$ and emulates $\mathcal{A}_1$'s key-generation oracle by using its own oracle.

  At some point $\mathcal{A}_1$ stops giving in output a bit b'. Then, $\mathcal{B}_1$ outputs $(b', b, \vec{x}_{1-b}, \mathcal{Q})$ as its own output, where $\mathcal{Q} = (\mathsf{k}_i)$ is the list of keys for which $\mathcal{A}$ has issued a key-generation query.

$\mathcal{D}(\mathsf{Mpk}, \vec{x}, \alpha)$:

- $\mathcal{D}$ interprets α as $\alpha = (b', b, \vec{x}_{1-b}, \mathcal{Q})$ and returns 1 if $b = b'$ and for any $k \in \mathcal{Q}$ $F(k, \vec{x}) = F(k, \vec{x}_{1-b})$, 0 otherwise.

Let $\mathsf{IND}_{\mathcal{A}}^{\mathsf{FE},b}$ be an experiment identical to the IND-Security experiment except that the challenger always encrypts challenge vector $\vec{x}_b$ (instead of choosing one of the two challenges at random). Then, with all but negligible probability, it holds that:

$$\mathsf{IND}_{\mathcal{A}}^{\mathsf{FE},0} = \mathsf{RealExp}^{\mathsf{FE},\mathcal{B}^0} \approx \mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^0} = \mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^1} \approx \mathsf{RealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^1} = \mathsf{IND}_{\mathcal{A}}^{\mathsf{FE},1} \ .$$

where:

1. $\mathsf{IND}_{\mathcal{A}}^{\mathsf{FE},0} = \mathsf{RealExp}^{\mathsf{FE},\mathcal{B}^0}$ holds by definition of $\mathcal{B}^0$ and $\mathcal{D}$.

2. $\mathsf{RealExp}^{\mathsf{FE},\mathcal{B}^0} \approx \mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^0}$ holds by the RSIM-Security of FE.

3. $\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^0} = \mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^1}$ holds because if $\mathcal{A}$ breaks the IND-Security of FE, then with all but negligible probability, the queries issued by $\mathcal{A}$ (and thus by $\mathcal{B}$) are such that $F(k, \vec{x}_0) = F(k, \vec{x}_1)$ for any key k for which $\mathcal{A}$ has issued a key-generation query.

4. $\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^1} \approx \mathsf{RealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^1}$ holds again by the RSIM-Security of FE.

5. Finally, $\mathsf{RealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{B}^1} = \mathsf{IND}_{\mathcal{A}}^{\mathsf{FE},1}$ holds by definition of $\mathcal{B}^0$ and $\mathcal{D}$.

But if $\mathsf{IND}_{\mathcal{A}}^{\mathsf{FE},0} \approx \mathsf{IND}_{\mathcal{A}}^{\mathsf{FE},1}$, then $\mathcal{A}$ does not break the IND-Security of FE, a contradiction. ▮

E Pre-image samplability

We review the definition of pre-image samplability functionalities introduced by [O'N10].

Definition E.1 [[O'N10]] Functionality $F : K \times M \to \Sigma$ is *pre-image sampleable* (PS, for short) if there exists a *sampler* algorithm Sam such that for all *PPT adversaries* $\mathcal{A}$,

$$\text{Prob}\left[\left(m, (k_i)_{i=1}^{l=\mathsf{poly}(\lambda)}\right) \leftarrow \mathcal{A}(1^\lambda);\ m' \leftarrow \mathsf{Sam}(1^\lambda, |m|, (k_i, F(k_i, m))_{i=1}^{l}) : F(k_i, m) = F(k_i, m') \text{ for } i = 1, \ldots, l\right] = 1 - \nu(\lambda)$$

for a negligible function ν.

In our positive results we use the following result.

Theorem E.2 [[O'N10]] The functionality inner-product over $\mathbb{Z}_2$[9] is PS.

[9] O'Neill proved this result for more general fields but we only need it for $\mathbb{Z}_2$.