On the Achievability of Simulation-Based Security for Functional Encryption

Angelo De Caro[1], Vincenzo Iovino[2], Abhishek Jain[3], Adam O'Neill[4], Omer Paneth[5], and Giuseppe Persiano[6]

[1] IBM Research Zurich, Switzerland, angelo.decaro@gmail.com
[2] University of Luxembourg, Luxembourg, vinciovino@gmail.com,
[3] Johns Hopkins University, USA abhishek@cs.jhu.edu,
[4] Georgetown University, USA amoneill@gmail.com,
[5] MIT, USA, omer@bu.edu,
[6] University of Salerno, giuper@gmail.com

Abstract. Recently, there has been rapid progress in the area of functional encryption (FE), in which a receiver with secret-key sk_y can compute from an encryption of x the value $F(x, y)$ for some functionality F. Two central open questions that remain are: (1) Can we construct FE secure under an indistinguishability-based (IND) security notion for general circuits? (2) To what extent can we achieve a simulation-based (SIM) security notion for FE? Indeed, it was previously shown that IND-security for FE is too weak for some functionalities [Boneh et al. – TCC'11, O'Neill – ePrint '10], but that there exist striking impossibility results for SIM-security [Boneh et al. – TCC'11, Agrawal et al. – ePrint 2012].

Our work establishes a connection between these questions by giving a compiler that transforms any IND-secure FE scheme for general circuits into one that is SIM-secure for general circuits.

- In the *random oracle model*, our resulting scheme is SIM-secure for an *unbounded* number of ciphertexts and key-derivation queries. We achieve this result by starting from an IND-secure FE scheme for general circuits with random oracle gates.
- In the *standard model*, our resulting scheme is secure for a bounded number of ciphertexts and non-adaptive key-derivation queries (i.e., those made before seeing the challenge ciphertexts), but an *unbounded* number of adaptive key-derivation queries. These parameters match the known impossibility results for SIM-secure FE and improve upon the parameters achieved by Gorbunov et al. [CRYPTO'12].

The techniques for our compiler are inspired by constructions of non-committing encryption [Nielsen – CRYPTO '02] and the celebrated Feige-Lapidot-Shamir paradigm [FOCS'90] for obtaining zero-knowledge proof systems from witness-indistinguishable proof systems.

Our compiler in the standard model requires an IND-secure FE scheme for general circuits, it leaves open the question of whether we can obtain SIM-secure FE for special cases of interest under weaker assumptions. To this end, we next show that our approach leads to a direct construction of SIM-secure *hidden vector encryption* (an important special case of FE that generalizes anonymous identity-based encryption). The scheme, which is set in composite order bilinear groups under subgroup decision assumptions, achieves security for a bounded number of ciphertexts but *unbounded* number of both non-adaptive and adaptive key-derivation queries, again matching the known impossibility results. In particular, to our knowledge this is the *first* construction of SIM-secure FE (for any non-trivial functionality) in the standard model handling an unbounded number of adaptive key-derivation queries.

Finally, we revisit the negative results for SIM-secure FE. We observe that the known results leave open the possibility of achieving SIM-security for various natural formulations of security (such as non-black-box simulation for non-adaptive adversaries). We settle these questions in the negative, thus providing essentially a full picture of the (un)achievability of SIM-security.

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

1 Introduction

One of the most important achievements of the modern theory of cryptography is the formalization of the notion of a secure cryptosystem given in the seminal work of Goldwasser and Micali [GM84]. There, two notions of security were proposed, one is an indistinguishability-based notion and the other is simulation-based, and they were shown to be equivalent in the sense that an encryption scheme meets one security definition if and only if it meets the other.

More than twenty years onwards, researchers have started looking at a much more sophisticated type of encryption called *functional encryption* (FE). A *functionality* F is a function $F : K \times M \to \{0,1\}$ where K is the *key space* and M is the *message space*. A *functional encryption scheme for* F is a special encryption scheme in which, for every *key* $k \in K$, the owner of the secret key Msk associated with the public key Pk can generate a special token or secret key $\mathtt{Tok}_k$ that allows the computation of $F(k, m)$ from a ciphertext of m computed under public key Pk. In other words, whereas in traditional encryption schemes decryption is an all-or-nothing affair, in functional encryption it is possible to finely control the amount of information that is revealed by a ciphertext. This opens up exciting applications to access control, executing search queries on encrypted data, and secure delegation of computation, among others.

Unlike in the case of classical cryptosystems, a general study of the security of FE did not appear initially. Instead, progressively more expressive forms of FE were constructed in a series of works (see, e.g., [BDOP04,BW07,KSW08,LOS$^+$10,OT12,Wat12]) that adopted indistinguishability-based (IND) notions of security. 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]. Quite interestingly, they show there exists clearly insecure FE schemes for certain functionalities that are nonetheless deemed secure by IND security, whereas these schemes do not meet the stronger notion of SIM-security. On the other hand, negative results have also emerged showing SIM-security is not always achievable [BSW11,BO12,AGVW12]. This leads to the main question that we study in this work:

> To what extent is SIM-security for functional encryption achievable? In particular, can schemes for IND-secure functional encryption be "bootstrapped" to the stronger notion of SIM-security?

In other words, while previous work investigated whether IND-security implies SIM-security, we study the more general question of whether a SIM-secure FE scheme can be constructed from an IND-secure one. Our results demonstrate that, despite the weaknesses of IND, essentially optimal constructions of SIM-secure FE can indeed be constructed based on IND-secure ones. We also close some possibilities for achieving SIM-security left open by existing negative results by extending them to various alternative natural notions of simulation (such as non-black-box and unbounded simulation), thus providing essentially a full picture of the achievability of SIM-security. We next discuss our results in more detail.

1.1 A Compiler for General Functionalities

Our first result is a general transformation that takes an IND-secure functional encryption scheme for all polynomial-size circuits and constructs a functional encryption scheme for the same functionality that is SIM-secure. We actually give two transformations, one in the (programmable) random oracle model [BR93] and one in the standard model.

To design our transformations, we make a connection between the construction of SIM-secure FE from IND-secure FE and the construction of zero-knowledge proof systems from witness indistinguishable proof systems, as studied in the celebrated work of Feige, Lapidot and Shamir [FLS90]. Recall that in the FLS paradigm, the simulator operates the proof system in a "trapdoor" mode which is indistinguishable from the behavior of the honest party to the adversary.

Adopting this paradigm to FE, we define a notion of "trapdoor circuits" which have additional "slots" in plaintext and keys that are used only by the simulator to program the functionality, not by the real system. (That is, we increase the length of the plaintext and keys versus the starting FE scheme to include these extra slots.) More concretely, one such slot in the plaintext is a flag that indicates to a secret key that the system is operating in trapdoor mode. This tells the key to check its own additional slots as well as additional slots in the plaintext to determine the output of the functionality in some fashion. For example, in the random oracle model, we can mask the output value using a hash function, so that it can be adaptively programmed by the simulator using an extension to the technique of Nielsen [Nie02] for achieving non-committing encryption. Using this approach:

- In the *standard model*, we achieve SIM-security for a bounded number of ciphertexts and non-adaptive key-derivation queries (i.e., those made before seeing the challenge ciphertext) but an *unbounded* number of adaptive key-derivation queries.
- In the *random oracle model*, we achieve SIM-security for an *unbounded* number of challenge messages and key-derivation (i.e., token) queries, which is the best level of security one can hope for.

The result in the standard model exactly matches recent impossibility results (as well as improvement we give that are discussed later).[7] Specifically [BSW11,BO12] show that a bounded number of ciphertexts is necessary when considering adaptive queries. (And indeed, if we only consider non-adaptive queries, our construction can be extended to an unbounded number of challenge messages.) Furthermore, Agrawal *et al.* [AGVW12] show that a bounded number of *non-adaptive* key-derivation queries is necessary for general functionalities (that is, they give an impossibility result for a particular functionality).

FE in the random oracle model. In the random oracle model transformation, the trapdoor circuits we use must themselves query the random oracle. Therefore we require that the functionality supported by the IND-secure FE includes even circuits that access the random oracle. We also augment the definition of IND-secure FE to address functionalities that access an oracle. We note that this notion of FE in the random oracle model is incomparable to IND-secure FE in the standard model.

Comparison to Prior Constructions. We note that it is currently a central open question in functional encryption to construct an IND-secure FE scheme for general circuits.[8] Therefore, we do not currently obtain any concrete new construction as a result of our compiler. Rather, we see this result as providing a connection between this question and the question of whether SIM-secure FE can be achieved (saying that for general functionalities, achieving SIM security is *no harder* than achieving IND, so research can focus on the latter).

If IND-secure FE for general functionalities is achieved our compiler will give interesting new results. In the random oracle model, the only prior construction to achieve SIM-security (for an unbounded number of ciphertexts and token queries) is the identity-based encryption

[7] We note that our transformed scheme, as for all our constructions of SIM-secure FE in the standard model meeting bounded security notions (such as for HVE described below), still meets IND-security for an unbounded number of messages and key queries. In this sense, it achieves "hedged" security in the unbounded case.

[8] We emphasize that, since our transformation matches the known impossibility results, we do not obtain any impossible result for IND-secure FE. Indeed, we believe IND-secure FE for general circuits is possible.

scheme of Boneh *et al.* [BSW11]; in fact, they explicitly raise the open question of constructing SIM-secure FE for general functionalities in the random oracle model. In the standard model, the prior construction of Gorbunov *et al.* [GVW12] achieves SIM-security for only a bounded number of adaptive token queries, whereas we handle an *unbounded* number. Furthermore their scheme has ciphertext size depending on the size of the circuit for the functionality, whereas ours does not enforce this (i.e., it has this property only if the starting IND-secure scheme does). We note that Goldwasser *et al.* [GKP^{+}12] recently gave a construction where ciphertext size does not depend on the output size, but they handle only a bounded number of non-adaptive key queries. Finally, we mention recent works [SW12,GVW13] that construct attribute-based encryption (ABE) for general circuits (i.e., "public-index predicate encryption" in the terminology of [BSW11]). Unfortunately, ABE does not seem sufficient for our compiler (roughly, because we need a "private index" property to prevent the adversary from detecting the trapdoor mode).

Why is IND-security Enough? The above shows that, surprisingly, despite the weaknesses of IND-security shown in [BSW11,O'N10], an IND-secure FE scheme for general circuits is enough to go "all the way" to SIM security. A natural question is how this can be the case, given that IND-security is known to be weak (even vacuous) for some functions. To answer this, let us look at the counter-example provided by [O'N10], which is an f (which we think of here as a circuit) for which there is another function g such that g is "hard to compute" from f but is isomorphic to g, meaning f and g have the same equality pattern across the domain. In this case, despite IND-security, a token for f may also allow computing g. However, the corresponding transformed circuit produced by our compiler is such that it agrees with f (via its added programmable "slots" in the plaintext) on all the challenge messages but *no longer agrees with* g, because g is computed honestly on the actual encrypted message, which is a "dummy" one. Thus, the transformed circuit cannot have this weakness (since otherwise we could violate IND-security).

1.2 Simulation-Secure Hidden Vector Encryption

Note that our compiler requires a IND-secure FE scheme for all polynomial-time circuits; in general, for a specific functionality F, we do not know how to compile an IND-secure FE scheme for F into a SIM-secure one. However, we also show that our approach leads to a direct construction of hidden-vector encryption HVE, a generalization of anonymous IBE introduced by [BW07]. The scheme is set in composite order bilinear groups and is proven secure under the general subgroup decision assumption [BWY11]. Our scheme is similar to IND-secure constructions (see [OT12,DCIP13]) except that it uses additional subgroups. Indeed, the presence of an additional subgroup component in the simulated ciphertexts acts as the "flag" here.[9] The simulated secret key also has a component in this subgroup which we can use to program the output of the functionality to output 0 when needed (the output of HVE is only one bit). Our scheme is secure for a bounded number of challenge messages (which is necessary when considering adaptive queries in the standard model as per [BSW11,BO12]) and an unbounded number of key-derivation queries both before and after seeing the challenge ciphertext. (The impossibility result of [AGVW12] does not apply to HVE.) In particular, to the best of our knowledge our scheme is the *first* construction of SIM-secure FE (for any functionality) in the standard model handling an unbounded number of adaptive token queries.

We remark that a functional encryption scheme for HVE can be derived from the general result of [GVW12]. However, the construction of [GVW12] imposes an upper bound on the

[9] In this sense, we compile an IND-secure scheme into a SIM-secure scheme in a non-blackbox way; we are hopeful that this approach can work for schemes in composite order groups for other functionalities as well.

number of tokens that can be seen by an adversary and is less efficient (this is due to the fact that our construction is tailored for HVE whereas the one of [GVW12] is a for a general class). On the other hand, [O'N10] shows that when restricted to non-adaptive token queries, IND and SIM security are equivalent for "preimage sampleable" functionalities. Interestingly, we show that if HVE is pre-image sampleable then it is possible to decide $\mathbb{NP}$ in probabilistic polynomial time. Therefore it is unlikely that pre-image samplability can be used to construct SIM-secure HVE, and in any case this would achieve security only for non-adaptive token queries.[10]

1.3 Stronger Impossibility Results

The positive results presented above should be compared to the known impossibility results for simulation-secure FE. Boneh, Sahai and Waters [BSW11] show an impossibility for an adversary that makes an unbounded number of ciphertext queries and one *adaptive* token query. Recently, Agrawal, Gorbunov, Vaikuntanathan, and Wee [AGVW12] showed a different impossibility result for an adversary that make an unbounded number of *non-adaptive* token queries and one ciphertext query. To complete the picture and understand the limitation of simulation-based secure FE we address several natural notions of security that fall outside the reach of the existing impossibilities, but for which no positive results are known.

We first focus on the aspect of adaptivity in negative results above. We note that both of the results are established by arguing about the order in which the adversary's view is simulated. In particular, the definition considered by [AGVW12] mandates that the simulator first simulate the tokens and only then simulate the ciphertexts. Similarly, [BSW11] uses a non-programmable random oracle to guarantee that the simulator first simulate ciphertexts and only then simulate tokens (recently, [BO12] showed how to eliminate the random oracle from this negative result). We ask whether impossibilities exist even for natural security notions that do not enforce such ordering on the simulation strategy.

Concretely, the impossibility of [AGVW12] uses a *black-box* simulation-based security definition that enforces order on the simulator. This leaves open the possibility of of simulation secure FE against adversaries making unbounded non-adaptive key queries using non-black-box simulators. Indeed, in other contexts, non-black-box simulation has proved to be strictly more powerful then black-box simulation [Bar01]. As another example, the impossibility of [AGVW12] addresses the case of non-adaptive key queries, however, it relies on the fact that the ciphertext query is made *after* key queries are answered. Thus, we consider the natural question of achieving simulation-secure FE against adversaries that are *fully non-adaptive* in the sense that they choose their ciphertext and token queries simultaneously (considered for the first time in this work).

We give the appropriate security definitions both for non-black-box simulation with non-adaptive key queries (based on the non-black-box definition of [BSW11]), and for simulation with fully non-adaptive adversaries. We then show that both definitions are not achievable. For the case of non-adaptive key queries we demonstrate this by using the techniques of [BO12] in the context of the [AGVW12] impossibility. For the fully non-adaptive case, we give a new extension of [AGVW12] that uses unbounded number of ciphertext queries as well as key queries.

Our second focus is on the output length of the functionality. Recently [GKP^{+}12] give a construction of FE for boolean functionalities with "succinct" ciphertexts. That is, ciphertext

[10] We note that a couple recent works [BO12,BF12] extend the equivalence of IND and SIM for pre-image sampleable functionalities to adversaries that also make very restricted adaptive key queries (but an unbounded number of them); however, in this work we are only interested in potentially bounding number of such queries but not restricting them in any other way.

length does not grow with the size of the functionality's circuit, but only with the depth. In contrast, for functionalities with longer output, the ciphertext size in all existing constructions grows linearly with the output length of the functionality. It is natural to wonder if we can construct an FE scheme where the ciphertext size is instead *independent* of the functionality's output length. Such a scheme is desirable since it does not force the encrypting party to place an a priori upper bound on the output length of the functions that can be evaluated from the ciphertext. Additionally, such an encryption scheme could potentially support functionalities that perform some computation on the plaintext and output a new FE ciphertext encrypting the result. The possibility of combining such functionalities and evaluating them recursively on their own output, has various applications [AGG12].

Unfortunately, we rule out such schemes in general. Namely, we give syntax for such FE (previous definition implicitly bound the functionality's output length) and show that such FE cannot satisfy the simulation-based security definition. Our impossibility uses ideas similar to [AGVW12], however we do not assume unbounded collusion. In fact, our impossibility holds for adversaries that make only single ciphertext query and a single and non-adaptive key query.

We summarize the existing and new negative results for simulation-secure FE in Table 1.

Work	Simulation type	NA-Key queries	Ad-Key queries	Simultaneous Key queries	Ciphertext queries	Simulation time	Short ciphertext?
[BSW11]	NBB (NPRO Model)	-NA-	2	-NA-	UB	UB	-NA-
[BO12]	NBB	-NA-	2	-NA-	UB	PPT	-NA-
[AGVW12]	BB	UB	-NA-	-NA-	1	PPT	-NA-
This	NBB	UB	-NA-	-NA-	1	PPT	-NA-
This	NBB	-NA-	-NA-	UB	UB	PPT	-NA-
This	NBB	-NA-	UB	-NA-	UB	UB	-NA-
This	NBB	-NA-	1	-NA-	1	PPT	Yes

Table 1. Negative Results for Sim-secure FE. UB stands for unbounded number of queries or running time. BB/NBB stands for Black-box/Non-black-box simulation.

1.4 Organization

In Section 2 we give the basic definitions for FE. In Section 3 we describe the transformations from IND-secure to SIM-secure FE, both in the random oracle model and in the plain model. Section 4 describes the construction of SIM-secure FE for the hidden vector encryption functionality. The details of the negative results are given in Section 5.

1.5 Subsequent Work

Subsequent to our work, [GGH$^+$13] constructed the first (selective) IND-secure FE scheme for general circuits based on indistinguishability obfuscation [BGI$^+$01,GGH$^+$13]. Subsequently, their construction was extended to achieve adaptive security in multiple works based on varying assumptions [Wat15,ABSV15,GGHZ16]. By applying our transformation from IND-security to SIM-security in the standard model, these results can be extended to obtain SIM-secure FE.

In a subsequent work, Iovino and Żebrowski [IŻ15] constructed a SIM-secure FE scheme in the random oracle model where the number of (non-adaptive and adaptive) key queries are a priori bounded. The main advantage of their result is that the bound on the key queries is not reflected in size of the ciphertexts but only on the decryption time.

Hubacek and Wichs [HW15] give a general lower bound on the communication complexity of secure function evaluation for functions with long output. Some of the known impossibility results for SIM-secure functional encryption follow as corollaries from their result.

In an attempt to overcome some existing impossibility results, De Caro and Iovino [CI16] proposed a new simulation-based security definition where the simulator can rewind the the adversary.

Recently, Agrawal, Koppula and Waters [AKW16] proved that there does not exist any $(0, \mathsf{poly}, \mathsf{poly})$-SIM-secure FE scheme for the class of (weak) pseudo-random functions in the random oracle model. This does not contradict our transformation in the random oracle model as it relies upon an IND-secure scheme for boolean circuits with random oracle gates. Indeed, (when combined with our transformation) their result rules out the existence of such an IND-secure FE scheme.

2 Definitions

A *negligible* function $\mathsf{negl}(k)$ is a function that is smaller than the inverse of any polynomial in k. If D is a probability distribution, the writing "$x \leftarrow D$" means that x is chosen according to D. If D is a finite set, the writing "$x \leftarrow D$" means that x is chosen according to uniform probability on D. If $q > 0$ is an integer then $[q]$ denotes the set $\{1, \ldots, q\}$. All algorithms, unless explicitly noted, are probabilistic polynomial time and all adversaries and distinguishers are modeled by non-uniform polynomial time algorithms. If B is an algorithm and A is an algorithm with access to an oracle then A^B denotes the execution of A with oracle access to B. If a and b are arbitrary strings, then $a|b$ denotes the string representing their delimited concatenation.

We now present some basic definitions that we will later use in our formulation of functional encryption.

Functionality. We start by defining the notion of a functionality.

Definition 1 [Functionality] A *functionality* $F = \{F_n\}_{n>0}$ is a family of functions $F_n : K_n \times M_n \to \Sigma$ where K_n is the *key space* for parameter n, M_n is the *message space* for parameter n and Σ is the *output space*. Sometimes we will refer to functionality F as a function from $F : K \times M \to \Sigma$ with $K = \cup_n K_n$ and $M = \cup_n M_n$.

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

Definition 2 [Circuit Functionality] The Circuit functionality has key space K_n equals to the set of all n-input Boolean circuits $\mathcal{C}_n$ and message space M_n the set $\{0,1\}^n$ of n-bit strings. For $C \in \mathcal{C}_n$ and $m \in M_n$, we have

$$\mathsf{Circuit}(C, m) = C(m),$$

2.1 Functional Encryption in the Standard Model

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

Below, we present the syntax, correctness, and indistinguishability and simulation-based security definitions of functional encryption in the standard model. In Section 2.2, we extend these definitions to the random oracle model.

Syntax. We start by presenting the syntax of a functional encryption scheme. A functional encryption scheme FE for the circuit functionality $\mathsf{Circuit}$ is a tuple $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ of four PPT algorithms:

1. $\mathsf{Setup}(1^\lambda, 1^n)$ outputs public and master secret keys $(\mathsf{Pk}, \mathsf{Msk})$ for security parameter λ and length parameter n that are polynomially related.
2. $\mathsf{KeyGen}(\mathsf{Msk}, C)$, on input a master secret key Msk and a circuit $C \in \mathcal{C}_n$ outputs a token $\mathtt{Tok}$.
3. $\mathsf{Enc}(\mathsf{Pk}, m)$, on input public key Pk and plaintext $m \in M_n$ outputs ciphertext Ct;
4. $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$ outputs $y \in \Sigma \cup \{\perp\}$.

Correctness. For all $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, all $C \in \mathcal{C}_n$ and $m \in M_n$, for $\mathtt{Tok} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, C)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, m)$, we have that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}) = F(k, m)$ whenever $\mathsf{Circuit}(C, m) \neq \perp$, except with negligible probability over the coins of KeyGen, Enc and Eval.

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

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

1. Ch generates $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and runs $\mathcal{A}_0$ on input Pk;
2. $\mathcal{A}_0$ submits queries $C_i \in \mathcal{C}_n$ for $i = 1, \ldots, q_1$ and, for each such query, Ch computes $\mathtt{Tok}_i = \mathsf{FE.KeyGen}(\mathsf{Msk}, C_i)$ and sends it to $\mathcal{A}_0$.
   When $\mathcal{A}_0$ stops, it outputs two *challenge plaintexts* $m_0, m_1 \in M_n$ and its internal state $\mathtt{st}$.
3. Ch picks $b \in \{0, 1\}$ at random, computes the *challenge ciphertext* $\mathsf{Ct} = \mathsf{Enc}(\mathsf{Pk}, m_b)$ and sends Ct to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$.
4. $\mathcal{A}_1$ submits queries $C_i \in \mathcal{C}_n$ for $i = q_1 + 1, \ldots, q$ and, for each such query, Ch computes $\mathtt{Tok}_i = \mathsf{KeyGen}(\mathsf{Msk}, C_i)$ and sends it to $\mathcal{A}_1$.
5. When $\mathcal{A}_1$ stops, it outputs b'.
6. **Output:** Output 1 if
   - $|m_0| = |m_1|$
   - For $i = 1 \ldots, q$, $C_i(m_0) = C_i(m_1)$
   - $b = b'$

   else output 0.

The advantage of adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ in the above game is defined as

$$\mathsf{Adv}^{\mathsf{FE},\mathsf{IND}}_{\mathcal{A}}(1^\lambda, 1^n) = \Pr[\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1^n) = 1] - 1/2$$

Definition 3 We say that FE is *indistinguishably secure* (IND security, for short) if all non-uniform PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ have at most negligible advantage in the above game.

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

Definition 4 [Simulation-Based security] A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for the circuit functionality $\mathsf{Circuit}$ defined over $(\mathcal{C}_n, M_n)$ is *simulation-secure* (SIM security, for short) if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for all *adversary* algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ the outputs of the following two experiments are computationally indistinguishable.

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

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

Remark 5 [Simulated Setup] The above follows the security definition of [GVW12] in that in the ideal experiment, the setup and non-adaptive key derivation queries are handled honestly (not by the simulator). More generally, we could have an algorithm $\mathsf{Sim.Setup}$ in place of $\mathsf{FE.Setup}$ in the ideal experiment; that is, the public key parameters of the ideal experiment could be generated by the simulator.

For simplicity, we use this relaxation with $\mathsf{Sim.Setup}$ in our construction for hidden vector encryption (and thus for IBE). (However, the construction can be modified to work with the above definition as well.) We stress that, in terms of security guarantees, however, we do not find a reason to prefer this more restrictive formulation.

Remark 6 [(q_1, q_2, ℓ)-SIM — Bounded messages and queries] To make the definition more precise, we define (q_1, q_2, ℓ)-SIM security, where $q_1 = q_1(\lambda), q_2 = q_2(\lambda), \ell = \ell(\lambda)$ are polynomials that are fixed a priori, as follows. In this definition, we require that SIM-security as define above holds for adversaries $\mathcal{A}$ where $\mathcal{A}_0$ makes at most q_1 queries and outputs message vectors of length and most ℓ, and furthermore $\mathcal{A}_1$ makes at most q_2 queries. In the case that a parameter is an unbounded polynomial we use the notation poly. Thus, for example, $(q_1, \mathsf{poly}, \ell)$-SIM security means that the adversary in the security definition makes a q_1-bounded number of non-adaptive key derivation queries but an unbounded number of adaptive key-derivation queries, and outputs a ℓ-bounded message vector. In the case that $q_1 = 0$ we say that the adversary is *strictly adaptive*.[11]

2.2 Functional Encryption in the Random Oracle Model

For one of our results, we will also consider functional encryption in the random oracle model.

[11] Curiously, due to [AGVW12], we see that a strictly adaptive adversary can be easier to handle than a non-adaptive one for general functionalities, in that we can achieve security for a bounded number of ciphertexts and unbounded number of adaptive queries, but the number of non-adaptive queries must be a-priori bounded.

Oracle-Aided Circuit Functionality. We consider oracle-aided boolean circuits that are given access a random oracle. Such a circuit queries to the random oracle and obtain answers that are then used in the computation of the circuit. One may also think of such a circuit as having random oracle gates.

We denote a random oracle by RO. A boolean circuit with random oracle gates is denoted as C^{RO}. Finally, the circuit functionality in the random oracle model is denoted as $\mathsf{Circuit}^{\mathsf{RO}}$.

Syntax. Let RO denote a random oracle. We consider functional encryption schemes for the $\mathsf{Circuit}^{\mathsf{RO}}$ functionality. Such a functional encryption scheme consists of four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ that are described in the same manner as in Section 2.1, with the following two differences: (1) The KeyGen algorithm takes as input a circuit C^{RO} (from $\mathsf{Circuit}^{\mathsf{RO}}$) with random oracle gates. (2) Further, all algorithms get access to the random oracle.

Correctness. For every oracle RO, every $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}^{\mathsf{RO}}(1^\lambda, 1^n)$, every n-input oracle-aided circuit C^{RO} in $\mathsf{Circuit}^{\mathsf{RO}}$, and every $m \in M_n$. Let $\mathtt{Tok} \leftarrow \mathsf{KeyGen}^{\mathsf{RO}}(\mathsf{Msk}, C^{\mathsf{RO}})$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}^{\mathsf{RO}}(\mathsf{Pk}, m)$. We have that $\mathsf{Eval}^{\mathsf{RO}}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}) = C^{\mathsf{RO}}(m)$ whenever $\mathsf{Circuit}^{\mathsf{RO}}(C, m) \neq \perp$, except with negligible probability over the coins of KeyGen, Enc and Eval.

Indistinguishability-based Security Indistinguishability-based security in the random oracle model is defined in a similar manner as the standard model except that the adversary gets access to the random oracle, and in order to win the game we require that in all the token queries, on both challenge plaintexts the circuit queried by the adversary produces the same output and makes the same sequence of oracle queries. It follows that the random oracle queries that occur during a token evaluation are not hidden.

Specifically let RO be a random oracle. We consider the game $\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}$ between an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* Ch described in Figure 1. We present the definition for only one message; it is easy to see the definition extends naturally for multiple messages.

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

1. Ch generates $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}^{\mathsf{RO}}(1^\lambda, 1^n)$ and runs $\mathcal{A}_0^{\mathsf{RO}}$ on input Pk;
2. $\mathcal{A}_0^{\mathsf{RO}}$ submits queries $C_i^{\mathsf{RO}} \in \mathcal{C}_n$ for $i = 1, \ldots, q_1$ and, for each such query, Ch computes $\mathtt{Tok}_i = \mathsf{FE.KeyGen}^{\mathsf{RO}}(\mathsf{Msk}, C_i^{\mathsf{RO}})$ and sends it to $\mathcal{A}_0^{\mathsf{RO}}$.
   When $\mathcal{A}_0$ stops, it outputs two *challenge plaintexts* $m_0, m_1 \in M_n$ and its internal state $\mathtt{st}$.
3. Ch picks $b \in \{0,1\}$ at random, computes the *challenge ciphertext* $\mathsf{Ct} = \mathsf{Enc}^{\mathsf{RO}}(\mathsf{Pk}, m_b)$ and sends Ct to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$.
4. $\mathcal{A}_1^{\mathsf{RO}}$ submits queries $C_i^{\mathsf{RO}} \in \mathcal{C}_n$ for $i = q_1 + 1, \ldots, q$ and, for each such query, $\mathsf{Ch}^{\mathsf{RO}}$ computes $\mathtt{Tok}_i = \mathsf{KeyGen}^{\mathsf{RO}}(\mathsf{Msk}, C_i^{\mathsf{RO}})$ and sends it to $\mathcal{A}_1^{\mathsf{RO}}$.
5. When $\mathcal{A}_1$ stops, it outputs b'.
6. **Output:** Output 1 if
   - $|m_0| = |m_1|$
   - For $i = 1, \ldots, q$, $C_i^{\mathsf{RO}}(m_0) = C_i^{\mathsf{RO}}(m_1)$ and the two executions $C_i^{\mathsf{RO}}(m_0), C_i^{\mathsf{RO}}(m_1)$ make exactly the same sequence of oracle queries.
   - $b = b'$

   else output 0.

Fig. 1. IND-security in RO Model.

The advantage of adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ in the above game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FE},\mathsf{IND}}(1^\lambda, 1^n) = \Pr[\mathsf{IND}_{\mathcal{A}}^{\mathsf{FE}}(1^\lambda, 1^n) = 1] - 1/2$$

Where the probability is also over RO.

Definition 7 We say that FE is *indistinguishably secure* (IND security, for short) if all non-uniform PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ have at most negligible advantage in the above game.

Simulation-based Security Simulation-based security in the random oracle model is defined in a similar manner as the standard model, except that all algorithms including the adversary, simulator and distinguisher (that tries to distinguish the output of the two experiments) get access to a random oracle RO. In the ideal experiment, we let the simulator "program" the random oracle's answers. In our application we will only consider simulation-based security in the random oracle model for the plain (non oracle-aided) circuit functionality.

Definition 8 A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for the circuit functionality $\mathsf{Circuit}$ is *simulation-secure in the programmable random oracle model* (SIM security in the RO model, for short) if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{SimRO})$ such that for all *adversary* algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, the outputs of the following two experiments are computationally indistinguishable.

```
$\mathsf{RealExp}^{\mathsf{FE},\mathcal{A}}(1^\lambda, 1^n)$

$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{FE.Setup}^{\mathsf{RO}}(1^\lambda, 1^n)$;
$(m, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{FE.KeyGen}^{\mathsf{RO}}(\mathsf{Msk},\cdot),\mathsf{RO}}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}^{\mathsf{RO}}(\mathsf{Pk}, m)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{FE.KeyGen}^{\mathsf{RO}}(\mathsf{Msk},\cdot),\mathsf{RO}}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;

Output: $(\mathsf{Pk}, m, \alpha)$
```

```
$\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{FE},\mathcal{A}}(1^\lambda, 1^n)$

$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{FE.Setup}^{\mathsf{SimRO}}(1^\lambda, 1^n)$;
$(m, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{FE.KeyGen}^{\mathsf{SimRO}}(\mathsf{Msk},\cdot),\mathsf{SimRO}}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim}_0(\mathsf{Pk}, |m|, \{C_i, \mathtt{Tok}_i, \mathsf{Circuit}(C_i, m)\})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot),\mathsf{SimRO}}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;

Output: $(\mathsf{Pk}, m, \alpha)$
```

Here, $\{C_i\}$ correspond to the token queries of the adversary. SimRO answers the random oracle queries made by any algorithm. Oracle $\mathcal{O}(\cdot)$ is the second stage of the simulator, namely algorithm $\mathsf{Sim}_1(\mathsf{Msk}, \mathsf{st}', \cdot, \cdot)$. Algorithm Sim_1 receives as third argument a circuit C_j for which the adversary queries a token, and as fourth argument the output value $\mathsf{Circuit}(C_j, m)$. Finally, note that Sim_0, Sim_1 and SimRO are stateful algorithms that share a common state.

3 From Indistinguishability to Simulation-Based Security

In this section, we show transformations from IND-secure functional encryption to SIM-secure functional encryption scheme.

In the random oracle model, the resulting scheme is secure for an unbounded number of messages and unbounded number of key queries, and in the standard model it is secure for a bounded number of messages and bounded number of non-adaptive key queries (but unbounded number of adaptive key queries). This matches known impossibility results, which we extend to even weaker security models in Section 5.

3.1 Trapdoor Circuits

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

Definition 9 [RO-Based Trapdoor Circuit] Let C be a circuit on n-bits. Let $\mathsf{RO}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a random oracle and $\mathcal{F} = \{f_s : s \in \{0,1\}^k\}_{k\in\mathbb{N}}$ be a pseudo-random function family. For $y \in \{0,1\}^n$ define the corresponding *RO-based trapdoor circuit* $\mathsf{Trap}_1^{\mathsf{RO}}[C,\mathcal{F}]^y$ on $(3n+1)$-bits as follows:

```
**Circuit** $\mathsf{Trap}_1^{\mathsf{RO}}[C,\mathcal{F}]^y(m')$
    $(m, \mathsf{flag}, x, r) \leftarrow m'$
    $z \leftarrow \mathsf{RO}(x, y)$
    If $\mathsf{flag} = 1$ then return $z \oplus f_r(y)$
    Else return $C(m)$
```

Definition 10 [Standard-Model Trapdoor Circuit] Fix $q > 0$. Let C be a n-bit input n-bit output circuit and let $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ be a symmetric-key encryption scheme with key-space $\{0,1\}^s$, message-space $\{0,1\}^n$, and ciphertext-space $\{0,1\}^\nu$. For $k' \in \{0,1\}^{n+\nu}$ define the corresponding *standard-model trapdoor circuit* $\mathsf{Trap}_2[C,\mathsf{SE}]^{k'}$ on $((2q+1)n+1+s)$-bit inputs and n-bit outputs as follows:

```
**Circuit** $\mathsf{Trap}_2[C,\mathsf{SE}]^{k'}(m')$
    $(r, c_{\mathsf{SE}}) \leftarrow k'$ ; $(m, \mathsf{flag}, k_{\mathsf{SE}}, (r_1,y_1),\ldots,(r_q,y_q)) \leftarrow m'$
    If $\mathsf{flag} = 1$ then
        If there exists $i$ such that $r = r_i$ then return $y_i$
        Else $y \leftarrow \mathsf{SE.Dec}(k_{\mathsf{SE}}, c_{\mathsf{SE}})$ ; Return $y$
    Else return $C(m)$
```

3.2 RO-based Transformation

Overview. We now present our transformation from IND-secure FE to SIM-secure FE in the random oracle model. Namely, we show how to transform an IND-secure FE scheme for the $\mathsf{Circuit}^{\mathsf{RO}}$ functionality (where circuits have random oracle gates) to a SIM-secure FE scheme for the $\mathsf{Circuit}$ functionality, that supports an unbounded number of message and token queries.

The main idea is to put additional "slots" in the plaintexts and secret keys that will only be used by the simulator. A plaintext will have four slots and a secret key will have two. In the plaintext, the first slot will be the actual message m. The second slot will be a bit flag indicating whether the ciphertext is in trapdoor mode. The third slot is a random string x and finally the fourth slot will be a seed r for a PRF. In the secret key, the first slot will be the actual circuit C and the second slot will be a random string y.

For evaluation, if the ciphertext is not in trapdoor mode ($\mathsf{flag} = 0$) then we simply output the result of the circuit C on the message m. If the ciphertext is in trapdoor mode, then the output is $\mathsf{RO}(x,y) \oplus f_r(y)$, where RO is a random oracle and f_r is the PRF keyed by r.

Definition 11 [RO-Based Transformation] Let $\mathsf{RO}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a random oracle and $\mathcal{F} = \{f_s : s \in \{0,1\}^k\}_{k\in\mathbb{N}}$ be a pseudo-random function family. Let $\mathsf{IndFE} = (\mathsf{IndFE.Setup}, \mathsf{IndFE.Enc}, \mathsf{IndFE.KeyGen}, \mathsf{IndFE.Eval})$ be a functional encryption scheme for the functionality $\mathsf{Circuit}^{\mathsf{RO}}$.

We define a new functional encryption scheme $\mathsf{SimFE}_1[\mathsf{RO}, \mathcal{F}] = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for $\mathsf{Circuit}$ as follows.

– $\mathsf{Setup}^{\mathsf{RO}}(1^\lambda, 1^n)$: returns the output of $\mathsf{IndFE.Setup}^{\mathsf{RO}}(1^\lambda, 1^{3n+1})$ as its own output.
– $\mathsf{Enc}^{\mathsf{RO}}(\mathsf{Pk}, m)$: on input Pk and $m \in \{0,1\}^n$, the algorithm chooses x at random from $\{0,1\}^n$, sets $m' = (m, 0, x, 0^n)$ and returns $\mathsf{IndFE.Enc}^{\mathsf{RO}}(\mathsf{Pk}, m')$ as its own output.
– $\mathsf{KeyGen}^{\mathsf{RO}}(\mathsf{Msk}, C)$: on input Msk and a n-input Boolean circuit C, the algorithm chooses random $y \in \{0,1\}^n$ and returns $(y, \mathtt{Tok})$ where $\mathtt{Tok} \leftarrow \mathsf{IndFE.KeyGen}^{\mathsf{RO}}(\mathsf{Msk}, \mathsf{Trap}_1^{\mathsf{RO}}[C, \mathcal{F}]^y)$.
– $\mathsf{Eval}^{\mathsf{RO}}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$: on input Pk, Ct and $\mathtt{Tok}$, returns the output $\mathsf{IndFE.Eval}^{\mathsf{RO}}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$.

Theorem 12 Suppose IndFE is an IND-Secure FE in the random oracle model for the $\mathsf{Circuit}^{\mathsf{RO}}$ functionality. Then SimFE_1 is an $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-SIM-secure FE in the random oracle model for the $\mathsf{Circuit}$ functionality.

Intuition. To gain some intuition, it is instructive to consider a simpler system where $f_r(y)$ in the evaluation is simply replaced by r. In this case, the fourth slot in the plaintext acts as an encryption under Nielsen's RO-based non-committing encryption scheme [Nie02], whose decryption can be adaptively programmed. However, this approach does not work for multiple keys, since then the simulator would need to program two hash outputs to $r \oplus C_1(m)$ and $r \oplus C_2(m)$, which would not look independently random to the distinguisher. To solve this problem, we use a PRF to generate a "fresh" ciphertext for each secret key. Since the number of secret keys is unbounded, we need to generate more randomness than can be contained in the plaintext slot, and thus we use a PRF.

Proof. Suppose there is an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against SimFE_1 that outputs at most $\ell = \ell(\lambda)$ messages and $q = q(\lambda)$ key-derivation queries. Note that here ℓ, q are unbounded polynomial, not fixed a priori. We construct a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{SimRO})$ as follows.

– First, $\mathsf{Setup}^{\mathsf{RO}}(1^\lambda, 1^n)$ computes $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{IndFE.Setup}^{\mathsf{RO}}(1^\lambda, 1^{3n+1})$.
– Let $m_1, \ldots, m_\ell$ be the challenge messages output by $\mathcal{A}_0$ on input Pk.
  Sim_0 receives as input the public parameter Pk, the q_1 non-adaptive token queries $C_1, \ldots, C_{q_1}$ made by $\mathcal{A}_0$, along with the values $z_{i,1} = C_1(m_i), \ldots, z_{i,q_1} = C_{q_1}(m_i)$ for each $1 \le i \le \ell$ and the tokens $(y_1, \mathtt{Tok}_1), \ldots, (y_{q_1}, \mathtt{Tok}_{q_1})$ generated by $\mathsf{KeyGen}^{\mathsf{RO}}$ to answer $\mathcal{A}_0$'s non-adaptive token queries. Sim_0 proceeds as follows:
  • For each $1 \le i \le \ell$, Sim_0 chooses random $x_i, r_i \in \{0,1\}^n$. If $\mathcal{A}_0$ queried the random oracle on any point (x_i, y_j) or if for any distinct $1 \le i, i' \le \ell$, $x_i = x_{i'}$ or if for any distinct $1 \le j, j' \le q_1$, $y_j = y_{j'}$, Sim_0 aborts.
  • For each $1 \le i \le \ell$ and for each $1 \le j \le q_1$, SimRO programs the random oracle by setting $\mathsf{RO}(x_i, y_j) = f_{r_i}(y_j) \oplus z_{i,j}$.
  • For each $1 \le i \le \ell$, Sim_0 sets $\tilde{m}_i = (0^n, 1, x_i, r_i)$ and then it computes $\tilde{\mathsf{Ct}}_i \leftarrow \mathsf{IndFE.Enc}^{\mathsf{RO}}(\mathsf{Pk}, \tilde{m}_i)$.
  Sim_0 outputs $(\tilde{\mathsf{Ct}}_1, \ldots, \tilde{\mathsf{Ct}}_\ell)$.
– Sim_1 answers the adaptive query for circuit C_j, for $j = q_1 + 1, \ldots, q$, by having on input the master secret key Msk and $z_{i,j} = C_j(m_i)$, for all $1 \le i \le \ell$, in the following way:
  • Sim_1 chooses random $y_j \in \{0,1\}^n$. If $\mathcal{A}_0$ or $\mathcal{A}_1$ queried the random oracle on any point (x_i, y_j) for $1 \le i \le \ell$ or if $y_j = y_{j'}$ for $1 \le j' \le j$, Sim_1 aborts.

- SimRO programs the random oracle by setting $\mathsf{RO}(x_i, y_j) = f_{r_i}(y_j) \oplus z_{i,j}$, for all $1 \le i \le \ell$. Sim_1 outputs the pair $(y_j, \mathsf{IndFE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}_1^{\mathsf{RO}}[C_j, F]^{y_j}))$.

All queries to RO are answered randomly except the queries that were programmed by SimRO. We now prove that Sim is a good simulator meaning that, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$ and $\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{SimFE},\mathcal{A}}$ are computationally indistinguishable.

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

– Hybrid $H_1^{\mathcal{A}}$: This is the ideal experiment $\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathsf{SimFE},\mathcal{A}}$.
– Hybrid $H_2^{\mathcal{A}}$: We change the way challenge ciphertexts are generated. Namely, instead of setting $\tilde{m}_i = (0^n, 1, x_i, r_i)$ to form challenge ciphertext $\tilde{\mathsf{Ct}}_i$, we set $\tilde{m}_i = (m_i, 0, x_i, 0^n)$.
– Hybrid $H_3^{\mathcal{A}}$: In this experiment, we change the strings $f_{r_i}(y_j)$ to independent truly random strings $r'_{i,j}$. That is, we program $\mathsf{RO}(x_i, y_j)$ to answer $r'_{i,j} \oplus z_{i,j}$ instead of $f_{r_i}(y_j) \oplus z_{i,j}$.
– Hybrid $H_4^{\mathcal{A}}$: This experiment never aborts and we change all answers given by RO to be truly random and independent of all other values in the experiment. This is the real experiment $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$.

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

Hybrid $H_1^{\mathcal{A}}$ to Hybrid $H_2^{\mathcal{A}}$: This transition reduces to IND-security of the starting FE scheme. For sake of contradiction, suppose there exists a distinguisher $\mathcal{D}$ that distinguishes with non-negligible probability the output distributions of $H_1^{\mathcal{A}}$ and $H_2^{\mathcal{A}}$. Then, $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful IND adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ for IndFE:

Adversary $\mathcal{B}_0$: On input the public parameters Pk generated by $\mathsf{IndFE.Setup}^{\mathsf{RO}}$, run $\mathcal{A}_0$ on input Pk forwarding its oracle queries to RO

– Answer $\mathcal{A}_0$'s key-derivation query C_i by using its oracle $\mathsf{IndFE.KeyGen}^{\mathsf{RO}}(\mathsf{Msk}, \cdot)$ as follows: choose a random $y_i \in \{0,1\}^n$ and output the pair $(y_i, \mathtt{Tok}_i)$ where $\mathtt{Tok}_i = \mathsf{IndFE.KeyGen}^{\mathsf{RO}}($ $\mathsf{Msk}, \mathsf{Trap}_1^{\mathsf{RO}}[C_i, \mathcal{F}]^{y_i})$.
– Eventually, $\mathcal{A}_0$ outputs message $m_1, \ldots, m_\ell$ and the state $\mathtt{st}$.
– For all $1 \le i \le \ell$, choose $x_i \in \{0,1\}^n$ and $r_i \in \{0,1\}^n$ at random. Then for each $1 \le i \le \ell$ and for each $1 \le j \le q_1$, program the random oracle by setting $\mathsf{RO}(x_i, y_j) = f_{r_i}(y_j) \oplus z_{i,j}$. Then set $\tilde{m}_i \leftarrow (0^n, 1, x_i, r_i)$, $\tilde{m}'_i = (m_i, 0, x_i, 0^n)$.
– Output $(\tilde{m}_1, \ldots, \tilde{m}_\ell)$ and $(\tilde{m}'_1, \ldots, \tilde{m}'_\ell)$ as the two challenge plaintexts.

Adversary $\mathcal{B}_1$: On input the public parameters Pk, a ciphertext Ct and the state $\mathtt{st}$, run $\mathcal{A}_1$ on the same inputs

– Answer a key-derivation query C_j made by $\mathcal{A}_1$ as follows: choose random $y_j \in \{0,1\}^n$ and program the random oracle by setting $\mathsf{RO}(x_i, y_j) = f_{r_i}(y_j) \oplus z_{i,j}$, for all $1 \le i \le \ell$. Return $(y_i, \mathtt{Tok}_i)$ where $\mathtt{Tok}_i \leftarrow \mathsf{IndFE.KeyGen}^{\mathsf{RO}}(\mathsf{Msk}, \mathsf{Trap}_1^{\mathsf{RO}}[C_i, \mathcal{F}]^{y_i}))$.
– Eventually, $\mathcal{A}_1$ outputs α.
– Invoke $\mathcal{D}$ on input $(\mathsf{Pk}, (m_1, \ldots, m_\ell), \alpha)$ and return $\mathcal{D}$'s guess.

We now bound the advantage of the IND adversary $\mathcal{B}$. First, we show that for every key-derivation query C_j made by $\mathcal{A}$, it is the case that for every for all $1 \le i \le \ell$, $\mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}(\tilde{m}_i)$, and $\mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}(\tilde{m}'_i)$ produce the same output and make the same oracle queries.

$$\begin{aligned}
\mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}(\tilde{m}_i) &= \mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}(m_i, 1, x_i, r_i) \\
&= \mathsf{RO}(x_i, y_i) \oplus f_{r_i}(y_i) \\
&= C_j(m_i) \\
&= \mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}(m_i, 0, x_i, 0^n) \\
&= \mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}(\tilde{m}'_i) \; .
\end{aligned}$$

Furthermore, from Definition 9, it follows that $\mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}$ makes the same random oracle queries on inputs $\tilde{m}_i$ and $\tilde{m}'_i$. In particular, $\mathsf{Trap}_1^{\mathsf{RO}}[C_j, \mathcal{F}]^{y_j}$ queries RO on input (x_i, y_i) in both cases.

Finally, $\mathcal{B}$ succeeds whenever $\mathcal{D}$ does.

Hybrid $H_2^{\mathcal{A}}$ to Hybrid $H_3^{\mathcal{A}}$: This transition reduces to security of the PRF. Since the distinguisher now gets no information about the PRF key, this reduction is straightforward and omitted here.

Hybrid $H_3^{\mathcal{A}}$ to $H_4^{\mathcal{A}}$: The probability that Sim_0 and Sim_1 sample some $(x_i, y_j) = (x_{i'}, y_{j'})$ where $(i, j) \neq (i', j')$ is negligible and since the probability that $\mathcal{A}_0$ or $\mathcal{A}_1$ query some (x_i, y_j) before either x_i or y_j is also negligible, the probability of abort in the experiment $H_3^{\mathcal{A}}$ is negligible. Since the values $r'_{i,j}$ are random and not used anywhere else in the experiment we have that the adversary's views in both experiments are statistically close.

3.3 Standard-Model Transformation

Overview. Here we give our transformation from IND to SIM security in the standard model. Again, we put additional "slots" in the plaintexts and secret keys that will only be used by the simulator. A plaintext will have $3 + 2q$ slots and a secret key will have two. As for the previous transformation, in the plaintext, the first slot will be the actual message m and the second slot will be a bit flag indicating whether the ciphertext is in trapdoor mode. Instead, the third slot will be a random symmetric key $\mathsf{sk}_{\mathsf{SE}}$, that will be used to handle adaptive queries, and the last $2q$ slots will be q pairs (r_i, z_i), where r_i is a random string and z_i is a programmed string. These $2q$ slots will be used to handle q non-adaptive queries. On the other hand, in a secret key for circuit C, the first slot is random string r, that for non-adaptive queries will be equal to one of the r_i in the challenge ciphertext, and c will be a ciphertext, under SE, for a programmed string.

For evaluation, if the ciphertext is not in trapdoor mode ($\mathsf{flag} = 0$) then we simply evaluate the original circuit C of the message m. If the ciphertext is in trapdoor mode, depending on the nature of the secret key (non-adaptive or adaptive), if $r = r_i$ for some $i \in [q]$ then we output z_i, otherwise we output $\mathsf{Dec}(\mathsf{sk}_{\mathsf{SE}}, \mathsf{c})$.

Definition 13 [Standard-Model Transformation] Let $\mathsf{IndFE} = (\mathsf{IndFE.Setup}, \mathsf{IndFE.Enc}, \mathsf{IndFE.KeyGen}, \mathsf{IndFE.Eval})$ be a functional encryption scheme for the functionality $\mathsf{Circuit}$. Let $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ be a symmetric-key encryption scheme with key-space $\{0,1\}^s$, message-space $\{0,1\}^n$, and ciphertext-space $\{0,1\}^\nu$. We require for simplicity that SE has pseudo-random ciphertexts.[12] We define a new functional encryption scheme $\mathsf{SimFE}_2[\mathsf{SE}] = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for $\mathsf{Circuit}$ as follows.

- $\mathsf{Setup}(1^\lambda, 1^n)$: returns the output of $\mathsf{IndFE.Setup}(1^\lambda, 1^{n(2q+1)+s+1})$ as its own output. In addition the algorithm picks a random key $\mathsf{sk}_{\mathsf{SE}} \in \{0,1\}^{\mathsf{s}}$ and keeps it in the secret key Msk.
- $\mathsf{Enc}(\mathsf{Pk}, m)$: on input Pk and $m \in \{0,1\}^n$, the algorithm sets $m' = (m, 0, 0^s, (0^n, 0^n), \ldots, (0^n, 0^n))$ and returns the output of $\mathsf{IndFE.Enc}(\mathsf{Pk}, m')$ as its own output.
- $\mathsf{KeyGen}(\mathsf{Msk}, C)$: on input Msk and a n-input Boolean circuit C, the algorithm chooses random r in $\{0,1\}^\lambda$, computes $c \in \{0,1\}^\nu$ as the encryption of a random n-bit plaintext with respect to $\mathsf{sk}_{\mathsf{SE}}$, computes $\mathtt{Tok}$ as $\mathtt{Tok} \leftarrow \mathsf{IndFE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}_2[C, \mathsf{SE}]^{k'})$ and returns $(r, c, \mathtt{Tok})$.
- $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$: on input Pk, ciphertext Ct and token $(r, c, \mathtt{Tok})$, returns the output of $\mathsf{IndFE.Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok})$.

[12] This assumption can easily be relaxed (by making simple modifications to our construction) to IND-CPA, but for practical blockcipher-based constructions it is not really an extra requirement anyway.

Theorem 14 Suppose IndFE is IND-Secure. Then SimFE_2 is $(q_1, \mathsf{poly}, 1)$-SIM-secure (cf. Remark 6).

We note that it is straightforward to modify our construction to be $(q_1, \mathsf{poly}, \ell)$-SIM secure (i.e., security for an a prior bounded number of challenge messages). The idea is simply to use many "lists" in the FE ciphertext and symmetric ciphertexts in FE keys. Furthermore, we remark that our construction is also IND-secure for an unbounded number of messages and key queries. In this sense, it can be viewed as having "hedged" security.

Intuition. The intuitions behind this theorem are very similar in spirit to that of Theorem 12. Let us restate them to clarify our choices. First, consider a simpler system where the $2q$ pairs (r_i, z_i) are replaced by a single pair $(\tilde{r}, \tilde{z})$. This approach does not work for multiple non-adaptive keys, since then the simulator would need to program $\tilde{z}$ to be $C_1(m)$ and $C_2(m)$ at the same time. To solve this problem, we add additional pairs in the ciphertext, one for each non-adaptive query. This is also the reason why we need an a-priori bound on the number of non-adaptive queries. For adaptive queries instead the simulator can program the second slot of the secret key to be an encryption of the expected output on the functionality.

Proof. Suppose there is an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against SimFE_2 that outputs at most q non-adaptive key queries. We construct a simulator $\mathsf{Sim} = (\mathsf{Sim.Setup}, \mathsf{Sim}_0, \mathsf{Sim}_1)$ as follows:

- $\mathsf{Sim.Setup}$ on inputs $1^\lambda, 1^n$ sets $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{IndFE.Setup}(1^\lambda, 1^{n(2q+1)+s+1})$.
- Sim_0 on input the public parameter Pk, token queries $C_1, \ldots, C_q$ made by $\mathcal{A}_0$, the Boolean values $z_1 = C_1(m), \ldots, z_q = C_q(m)$ where m is the challenge message output by the adversary, and the tokens $(r_1, c_1, \mathtt{Tok}_1), \ldots, (r_q, c_q, \mathtt{Tok}_q)$ generated by KeyGen to answers A_0's token queries, does the following:
  It chooses a random symmetric key k_{SE}. It sets $\tilde{m} = (0^n, 1, k_{\mathsf{SE}}, (r_1, z_1), \ldots, (r_q, z_q))$ and then it computes $\tilde{\mathsf{Ct}}$ as $\tilde{\mathsf{Ct}} \leftarrow \mathsf{IndFE.Enc}(\mathsf{Pk}, \tilde{m})$ and stores k_{SE} in the state.
- Sim_1 on input the master secret key Msk, the state, Boolean circuit C_j and $z_j = C_j(m)$ does the following:
  Sim_1 picks random $y_j \in \{0,1\}^n$ and then sets $c_j \overset{R}{\leftarrow} \mathsf{SE.Enc}(k_{\mathsf{SE}}, z_j)$, where k_{SE} is taken from the state, and $k'_j \leftarrow (y_j, c_j)$ and returns $\mathtt{Tok}_j$ as $(k'_j, \mathsf{IndFE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}_2[C, \mathsf{SE}]^{k'_j}))$.

We now prove that Sim is a good simulator meaning that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, $\mathsf{RealExp}^{\mathsf{SimFE}, \mathcal{A}}$ and $\mathsf{IdealExp}^{\mathsf{SimFE}, \mathcal{A}}_{\mathsf{Sim}}$ are computationally indistinguishable.

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

- Hybrid $H_1^{\mathcal{A}}$: This is the ideal experiment $\mathsf{IdealExp}^{\mathsf{SimFE}, \mathcal{A}}_{\mathsf{Sim}}$.
- Hybrid $H_2^{\mathcal{A}}$: In this experiment, we change how the challenge ciphertexts are generated. Instead of setting the challenge plaintext as $\tilde{m} = (0^n, 1, k_{\mathsf{SE}}, (r_1, z_1), \ldots, (r_q, z_q))$ we set $\tilde{m} = (m, 0, 0^\lambda, (0^n, 0^n), \ldots, (0^n, 0^n))$. This step reduces to IND security of the starting FE scheme.
- Hybrid $H_3^{\mathcal{A}}$: This is the real experiment $\mathsf{RealExp}^{\mathsf{SimFE}, \mathcal{A}}$. The main difference with $H_2^{\mathcal{A}}$ is that instead of setting, $c_j \overset{R}{\leftarrow} \mathsf{SE.Enc}(k_{\mathsf{SE}}, z_j)$, we set $c_j \overset{R}{\leftarrow} \{0,1\}^\nu$. This step reduces to pseudo-randomness of SE.

Hybrid $H_1^{\mathcal{A}}$ to Hybrid $H_2^{\mathcal{A}}$: For sake of contradiction, suppose there exists a distinguisher $\mathcal{D}$ that distinguishes with non-negligible probability the output distribution of $H_1^{\mathcal{A}}$ and $H_2^{\mathcal{A}}$. Then, $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful IND adversary $\mathcal{B}$ for IndFE. Specifically, $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ does the following.

- $\mathcal{B}_0$ on input the public parameters Pk generated by $\mathsf{IndFE.Setup}$, runs $\mathcal{A}_0$ on input Pk.
  Then, $\mathcal{B}_0$ answers $\mathcal{A}_0$'s key-derivation query C_i by using its oracle $\mathsf{IndFE.KeyGen}(\mathsf{Msk}, \cdot)$ as follows: $\mathcal{B}_0$ chooses random $r_i \in \{0,1\}^\lambda$ and $c_i \in \{0,1\}^\nu$ and sets $k'_i = (r_i, c_i)$, and returns $(k'_i, \mathtt{Tok})$ where $\mathtt{Tok} \leftarrow \mathsf{IndFE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}_2[C_i, \mathsf{SE}]^{k'})$.
  Eventually, $\mathcal{A}_0$ outputs a message m and the state $\mathtt{st}$. $\mathcal{B}_0$ chooses a random symmetric key k_{SE} and sets $x \leftarrow (0^n, 1, k_{\mathsf{SE}}, (r_1, z_1), \ldots, (r_q, z_q))$ and $x' \leftarrow (m, 0, 0^s, (0^n, 0^n), \ldots, (0^n, 0^n))$. $\mathcal{B}_0$ outputs x and x' as the two challenge plaintexts.
- $\mathcal{B}_1$ on input the public parameters Pk, a ciphertext Ct and the state $\mathtt{st}$, runs $\mathcal{A}_1$ on the same inputs.
  Then, $\mathcal{B}_1$ answers $\mathcal{A}_1$'s key-derivation query C_i as follows: $\mathcal{B}_1$ picks random $y_i \in \{0,1\}^n$ and then sets $c_i \overset{R}{\leftarrow} \mathsf{SE.Enc}(k_{\mathsf{SE}}, C_i(m))$, where k_{SE} and m are taken from the state, and $k'_i \leftarrow (y_i, c_i)$ and returns $\mathtt{Tok}_i$ as $(k'_i, \mathsf{IndFE.KeyGen}(\mathsf{Msk}, \mathsf{Trap}_2[C_i, \mathsf{SE}]^{k'_i}))$.
  Eventually, $\mathcal{A}_1$ outputs α, then $\mathcal{B}_1$ invokes $\mathcal{D}$ on input (Pk, m, α) and returns $\mathcal{D}$'s guess as its own guess.

Hybrid $H_2^{\mathcal{A}}$ to Hybrid $H_3^{\mathcal{A}}$: This transition reduces to the pseudo-randomness of SE. Since this transition is similar to the first transition and for lack of space, we omit the details here.

4 Simulation-Based Security for Hidden Vector Encryption

In the section we present a positive result for simulation-based secure functional encryption for HVE whose semantic security can be proved under static assumptions in the bilinear pairing setting in the standard model against *adaptive* adversaries obtaining a *bounded* number of ciphertexts and asking an *unbounded* number of tokens.

Definition 15 [HVE Functionality] The HVE functionality has message space M_n equal to the set of length n Boolean vectors $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ and key space K_n equal to the set length n Boolean vectors $\boldsymbol{y} = \langle y_1, \ldots, y_n \rangle$ with $\star$'s ("don't-care" entries). $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y})$ is equal to 1 iff, for all $1 \le i \le n$, $x_i = y_i$ or $y_i = \star$.

We stress that by the impossibility result of [BSW11] it is impossible to have simulation-based secure HVE against adversaries that see unbounded number of ciphertexts in the standard model. For simplicity, we present the case of one challenge ciphertext here. We use composite order bilinear groups whose order is the product of five distinct primes (see Appendix B for a description). Using the techniques of [Lew12], we can re-write our construction to work in prime order groups with only a constant blow-up of the number of group elements of ciphertexts and keys. We also note that one possible avenue for obtaining (non-adaptively) simulation-based secure functional encryption scheme for HVE could be via the notion of pre-image sampleability introduced by O'Neill [O'N10]. However, in Appendix A we prove that HVE is unlikely to be pre-image sampleable.[13]

4.1 The Scheme

In this section we describe our HVE scheme. To make our description and proofs simpler, we add to all vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ two dummy components and set both of them equal to 0. We can thus assume that all vectors have at least two non-star positions.

[13] On the other hand, it is easy to see that *every* functionality is pre-image sampleable if we allow unbounded simulation, which was recently considered by [AGVW12]. In this case, a SIM-secure scheme for HVE can be obtained directly from any IND-secure scheme (see [OT12,DCIP13]), but the security guarantees are only for non-adaptive adversaries and less clear.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4p_5, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization, and random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$, and, for $i \in [\ell]$ and $b \in \{0,1\}$, random $t_{i,b} \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets

$$T_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b} \ .$$

The public key is $\mathsf{Pk} = [\mathcal{I}, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$. The algorithm returns $(\mathsf{Pk}, \mathsf{Msk})$.

$\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$: Let $S_{\boldsymbol{y}}$ be the set of indices i such that $y_i \neq \star$. The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for $i \in S_{\boldsymbol{y}}$ under the constraint that $\sum_{i\in S_{\boldsymbol{y}}} a_i = 0$. For $i \in S_{\boldsymbol{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_{12}^{a_i/t_{i,y_i}} \cdot W_i \ .$$

The algorithm returns ciphertext $\mathsf{Ct} = (Y_i)_{i\in S_{\boldsymbol{y}}}$. Here we use the fact that $S_{\boldsymbol{y}}$ has size at least 2.

$\mathsf{Enc}(\mathsf{Pk}, \boldsymbol{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets

$$X_i = T_{i,x_i}^s \cdot Z_i \ ,$$

and returns the token $\mathtt{Tok}_{\boldsymbol{y}} = (X_i)_{i\in[\ell]}$.

$\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_{\boldsymbol{y}})$: The test algorithm computes $T = \prod_{i\in S_{\boldsymbol{y}}} \mathbf{e}(X_i, Y_i)$. It returns TRUE if $T = 1$, FALSE otherwise.

Correctness. It easy to see that the scheme is correct.

Theorem 16 Under the General Subgroup Decision Assumption (see Appendix B) the HVE scheme described in Section 4.1 is $(\mathsf{poly}, \mathsf{poly}, 1)$-SIM-secure (cf. Remark 6).

We note that one can easily extend our construction to meet $(\mathsf{poly}, \mathsf{poly}, \ell)$-SIM security for any a priori bounded polynomial ℓ. The idea is simply to use a different subgroup for each message in the "trapdoor" mode. We also note that our scheme can also be proven IND-secure for an unbounded number of messages and key queries. In this sense, it can be viewed as having "hedged" security.

Proof of Security. We give an overview of the proof and the full proof of security to Appendix C.

5 Impossibility results

In the section we present new negative results for simulation-based secure functional encryption. We first briefly recall the start of the art.

Boneh, Sahai and Waters [BSW11] show that for the IBE functionality, it is impossible to achieve (adaptive) simulation-based security when the adversary is allowed to make an unbounded number of ciphertext queries and one *adaptive* token query. Recently, Agrawal, Gorbunov, Vaikuntanathan, and Wee [AGVW12] showed a different impossibility result for the functionality that computes a weak pseudo-random function when the adversary is allowed to make an unbounded number of *non-adaptive* token queries and one ciphertext query. Both of the negative results above are established by arguing about the order in which the adversary's view is simulated. In particular, [AGVW12] make use of the argument that the simulator must first simulate the tokens and only then simulate the ciphertexts. Similarly, [BSW11] uses a non-programmable random oracle to enforce the ordering that the simulator must first simulate ciphertexts and only then simulate tokens. Recently, [BO12] show how to get rid of the random oracle in the result of [BSW11].

Our first contributions are for the non-adaptive case. In this case, [AGVW12] use a black-box simulation-based security definition that enforce the required order on the simulator (as discussed above). [AGVW12] therefore do not rule out the possibility of secure FE with non-black-box simulator or of FE that satisfies a relaxed security definition where the adversary makes all of it ciphertext and token queries simultaneously before getting any response from (we refer to this as the *fully-non-adaptive* simulation-based definition). By leveraging ideas from [BO12,AGVW12] we obtain new impossibility results for the above cases.

The second type of negative results we obtain are for FE for functionalities with "long" output. We note that in existing constructions of FE with simulation-based security, the size of the ciphertext grows linearly with the output length of the functionality [GVW13,GKP$^+$12]. We show that this is inherent by demonstrating an impossibility result for FE where the ciphertext length is independent of the functionality output length. We first modify the definition of [BSW11] slightly to allow for such independence (in prior definitions, the length of the functionality's output for every input length is fixed and the ciphertext length may depend on it). Then, we use techniques from [AGVW12] to get an impossibility for FE satisfying this definition. Unlike the impossibility of [AGVW12], our impossibility does not depend on the size of the collusion, and it holds even of the adversary only makes one token query and one ciphertext query. In particular, it also holds w.r.t. non-black-box simulation.

5.1 Non-Black-Box Definition

The simulation-based security definition for FE compares a real world experiment where the adversary interacts with the encryption scheme, to an ideal experiment where the view of the adversary is simulated by an efficient simulator. The definition of [AGVW12] considers only a simulator that is given black-box access to the adversary. Such a simulator must run the adversary, simulates answers to the adversary's queries, and output the adversary's view. In contrast, the impossibility of [BSW11] is with respect to a weaker definition where the simulator is given non-black-box access to the adversary. This definition puts no limitation on the simulation strategy. As we discuss shortly, the impossibility of [AGVW12] relies on the order in which the simulator answers the adversaries queries and therefore it does not hold for the non-black-box simulation-based security definition.

Next we present the definition for FE with non-black-box simulation. This definition is similar to the non-black-box definition of [BSW11] except for the following differences:

- We consider only non-adaptive key queries instead of only adaptive ones (this will suffice for our negative result). Since now the adversary may choose its message distribution to depend on received tokens, we cannot use the same massage sampling algorithm both in the real and in the ideal world. Instead, in the ideal world, the simulator will specify the message distribution.

– Following the definition of [BO12] we let the adversary and the simulator use an auxiliary input sampled from some distribution. In our negative result, we use this auxiliary input to store a random key of a collision-resistant hash function.

A stronger variant of FE with non-black-box simulation is defined in [BO12] and our negative result holds also for their definition.

Definition 17 [Non-Black-Box Simulation] A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, M) is simulation-secure with *non-black-box* simulator if for every distribution on auxiliary input Z, and every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that the outputs of the following two experiments are computationally indistinguishable:

```
$\mathsf{RealExp}^{\mathsf{FE},\mathcal{A}}(1^\lambda)$

$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$z \leftarrow Z$;
$(M, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk}, z)$;
$m \leftarrow M, \mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, m)$;
$\alpha \leftarrow \mathcal{A}_1(\mathsf{Ct}, \mathtt{st})$;
Let $\{k_i\}$ be the queries of $\mathcal{A}_0$ to $\mathsf{KeyGen}$;
**Output:** $(z, M, m, \alpha, \{k_i\})$
```

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

$z \leftarrow Z$;
$(M, \mathtt{st}) \leftarrow \mathsf{Sim}_0(z)$;
$m \leftarrow M$;
$\alpha \leftarrow \mathsf{Sim}_1^{F(\cdot,m)}(\mathtt{st})$;
Let $\{k_i\}$ be the queries of $\mathsf{Sim}_1$ to $F$;
**Output:** $(z, M, m, \alpha, \{k_i\})$
```

where the output of $\mathcal{A}_0$ and Sim_0 consists of an arbitrary state $\mathtt{st}$ and a description of a distribution over messages M.

The impossibility of [AGVW12]. We briefly recall the impossibility of [AGVW12] and explain why it does not hold for the non-black-box simulation-based security definition. For a pseudo-random function family $\{G_s\}_{s\in\{0,1\}^n}$ with one bit output, we consider the functionality $F(k, s) = G_s(k)$. Let $l - 1$ be an upper bound on the ciphertext size. The adversary asks tokens for l arbitrary keys $k_1, \ldots, k_l$ in the domain of G, and for an encryption of a random message s from the key space of G. The simulator needs to produce tokens $\{\mathtt{Tok}_i\}_{i\in[l]}$, and then it is given the functionality's outputs $\{G_s(k_i)\}_{i\in[l]}$. Now the simulator has to produce a ciphertext Ct such that for every $i \in [l]$, $G_s(k_i) = \mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_i)$. Now, on the one hand, the simulator needs to "encode" all of the functionality's outputs into Ct. On the other hand, the functionality's outputs are l pseudo-random bits, while $|\mathsf{Ct}| < l-1$. Since a pseudo-random string cannot be compressed by an efficient adversary we get a contradiction. (Note that the simulator cannot encode the functionality's outputs into the tokens $\{\mathtt{Tok}_i\}$ since these are fixed before the simulator learns the outputs.)

Impossibility of realizing the non-black-box definition (sketch). In the non-black-box simulation definition the real and the simulated outputs may contain the generated tokens and ciphertext. However, the simulator is only required to produce the simulated tokens and ciphertext after receiving the functionality's outputs. Since the tokens may encode a lot of information (at least as much as the functionality's outputs) the impossibility of [AGVW12] does not apply. To commit the simulator to the tokens before learning the functionality's outputs we use technique of [BDWY12]. This technique was recently used by [BO12] to extend the impossibility of [BSW11] to hold for a non-black-box definition and without using a non-programmable random oracle. The main idea is to use a collision-resistant hash function to commit the simulator to the tokens.

In order to extend the hash function idea to the non-adaptive setting, we consider a slightly different functionality that takes messages of the form (s, h) where s is a key for a the pseudo-random function family $G = \{G_s\}_{s\in\{0,1\}^n}$, and h is the output of a collision-resistant hash function taken from a family $H = \{H_i\}_{i\in\{0,1\}^n}$. The functionality $F(k,(s,h)) = G_s(k)$ computes the pseudo-random function ignoring the hash value. We define Z to be a distribution on auxiliary input that contains a random key for H. Consider the following real world adversary $(\mathcal{A}_0, \mathcal{A}_1)$. $\mathcal{A}_0$ is given a key i for H as auxiliary input. Let $l - 1$ be an upper bound on the ciphertext size. $\mathcal{A}_0$ will then make l token queries for arbitrary keys $k_1, \ldots, k_l$ in the domain of G. When $\mathcal{A}_0$ is given the tokens $\texttt{Tok} = (\texttt{Tok}_1, \ldots, \texttt{Tok}_l)$ it outputs the description of a distribution M on messages and it's entire state $\texttt{st}$. $\mathcal{A}_0$ computes $h_0 = H_i(\texttt{Tok}|\mathsf{Pk})$ and chooses M to be the uniform distribution over $\mathcal{U}_n \times h_0$. W.l.o.g assume that the distribution M is described simply by specifying h_0. That is, s is a random key for G and h is the constant h_0. $\mathcal{A}_1$ is given the ciphertext Ct, and the state of $\mathcal{A}_0$ including h_0 and the tokens $\texttt{Tok}$. $\mathcal{A}_1$ outputs $(\mathsf{Pk}, \mathsf{Ct}, h_0, \texttt{Tok})$. Let M be the distribution that $\mathcal{A}_0$ outputs. By construction we have that with probability 1 the real world experiment contains an index i, the message distribution M, a sampled message $(s, h) \leftarrow M$ and the output $(\mathsf{Pk}, \mathsf{Ct}, h_0, \texttt{Tok})$ such that M is specified by h_0, $h = h_0$ and for every $j \in [l]$, $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \texttt{Tok}_j) = G_s(k_j)$.

Assume towards contradiction that there exist a PPT simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ for $(\mathcal{A}_0, \mathcal{A}_1)$. By the correctness of the simulation it must be that with overwhelming probability over the auxiliary input i, Sim_0 outputs a state $\texttt{st}$ and the message distribution M specified by h_0. Additionally, with overwhelming probability over a message $(s, h_0) \leftarrow M$ and over Sim_1, $\mathsf{Sim}_1^{G_s(\cdot)}(\texttt{st})$ outputs $(\mathsf{Pk}', \mathsf{Ct}', h_0, \texttt{Tok}')$ such that $j \in [l]$, $\mathsf{Eval}(\mathsf{Pk}', \mathsf{Ct}', \texttt{Tok}'_j) = G_s(k_j)$.

We use Sim_1 to construct a distinguisher $\mathcal{D}$ that given oracle access to a function $\mathcal{O}$ can distinguish whether $\mathcal{O}$ is a random function in G or a truly random function. $\mathcal{D}$ runs Sim_1 on $\texttt{st}$ and forwards it oracle queries to $\mathcal{O}$. Sim_1 outputs $(\mathsf{Pk}', \mathsf{Ct}', h_0, \texttt{Tok}')$. $\mathcal{D}$ outputs 1 Iff $\mathsf{Eval}(\mathsf{Pk}', \mathsf{Ct}', \texttt{Tok}'_j) = \mathcal{O}(k_j)$ for all $j \in [l]$. As argued before, it follows from the correctness of the simulation that if $\mathcal{O}$ is a random function in G, $\mathcal{D}$ outputs 1 with overwhelming probability. To get a contradiction to the pseudo-randomness of G, it is left to bound the probability that $\mathcal{D}$ outputs 1 when $\mathcal{O}$ is a truly random function away from 1.

Consider a procedure Ch that samples $s \leftarrow \{0,1\}^n$ and runs Sim_1 on $\texttt{st}$ and with oracle access to G_s. With overwhelming probability, the output of Sim_1 contains a public key Pk_0 and tokens $\texttt{Tok}_0$ such that $h_0 = H_i(\texttt{Tok}_0|\mathsf{Pk}_0)$. Recall that in the execution of $\mathcal{D}$, if $h_0 \neq H_i(\texttt{Tok}'|\mathsf{Pk}')$ with noticeable probability then $\mathcal{D}$ outputs 0 with the same probability and we are done. If $h_0 = H_i(\texttt{Tok}', \mathsf{Pk}')$ then $\texttt{Tok}_0 = \texttt{Tok}'$ and $\mathsf{Pk}_0 = \mathsf{Pk}'$ with overwhelming probability, otherwise we can use $\mathcal{D}$ and Ch to construct an adversary for the collision-resistant hash. Since the values $\{\mathcal{O}(k_j)\}_{j\in[l]}$ are uniformly random and independent of $\texttt{Tok}_0, \mathsf{Pk}_0$ and since $|\mathsf{Ct}'| \leq l-1$ it follows that with probability 1/2 over the choice of $\mathcal{O}$, there does not exist Ct' such that $\mathsf{Eval}(\mathsf{Pk}_0, \mathsf{Ct}', \texttt{Tok}_0[j]) = \mathcal{O}(k_j)$ for all $j \in [l]$ and $\mathcal{D}$ must output 0. Thus, overall we get that if $\mathcal{O}$ is a truly random function, then $\mathcal{D}$ outputs 1 with probability at most negligibly over 1/2, which is a contradiction.

5.2 Fully Non-Adaptive Definition

In this section we give an impossibility result for a relaxation of the simulation-security considering only adversaries that are *fully non-adaptive*. Recall that prior security definitions consider the following two kinds of real world adversaries:

1. Non-adaptive adversaries who are only allowed to make token queries *before* the ciphertext queries.
2. Adaptive adversaries who, in addition to the above, are allowed to make additional token queries *after* receiving the ciphertexts, in an adaptive manner.

As discussed before, both the impossibility results in [BSW11,AGVW12] crucially rely on the order in which the adversaries queries are made. [BSW11] relies on the ability of the adversary to choose its token queries adaptively on the ciphertext and [AGVW12] use the fact that the adversary makes an *adaptive* ciphertext query after receiving the tokens. In contrast, we consider a natural relaxation of simulation-based security for FE where the real world adversary is strictly weaker than all previously considered definitions of simulation-based secure FE. In particular, we consider a *fully non-adaptive* adversary who makes token and ciphertext queries *simultaneously.* It is not difficult to see that the former impossibilities do not hold for such an adversary.

Below, we formally define security against fully non-adaptive adversaries. Our definition allows for non-black-box simulation. For simplicity of exposition, we define security for only one message. Our definition extends naturally to capture security for multiple messages.

Definition 18 [Fully Non-Adaptive Security] A functional encryption scheme FE for functionality F is *fully non-adaptively secure* if there exists a *simulator* algorithm Sim such that for all *adversary* algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ the outputs of the following two experiments are computationally indistinguishable.

$\mathsf{RealExp}^{\mathsf{FE},\mathcal{A}}(1^\lambda)$	$\mathsf{IdealExp}^{\mathsf{FE},\mathcal{A}}_{\mathsf{Sim}}(1^\lambda)$
$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;	$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(m, \{k_i\}_{i=1}^q, \mathtt{st}) \leftarrow \mathcal{A}_0(\mathsf{Pk})$;	$(m, \{k_i\}_{i=1}^q, \mathtt{st}) \leftarrow \mathcal{A}_0(\mathsf{Pk})$;
$\mathtt{Tok}_{k_i} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k_i)$;	$\alpha \leftarrow \mathsf{Sim}(\mathsf{Pk}, \mathsf{Msk}, \mathtt{st}, \{k_i, F(k_i, m)\})$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, m)$;	**Output:** (Pk, α)
$\alpha \leftarrow \mathcal{A}_1(\mathsf{Pk}, \{\mathtt{Tok}_{k_i}\}, \mathsf{Ct}, \mathtt{st})$;	
Output: (Pk, α)	

Impossibility of realizing the fully non-adaptive definition. We briefly sketch how to extend the impossibility of [AGVW12] for fully non-adaptive adversaries. The central idea is to use many ciphertext queries instead of just one. The intuition is that in the non-adaptive case the simulator can encode information about the function outputs in the tokens that might be long. By using many ciphertext queries, the same tokens are used to decrypt many ciphers, making the length of the tokens insignificant. Similarly to the impossibility of [AGVW12] we consider the functionality $F(k, s) = G_s(k)$ where $\{G_s\}_{s\in\{0,1\}^n}$ is a pseudo-random function family. Let λ be an a priori upper bounds on the length of the tokens and ciphertexts. For any $l > 2 \cdot \lambda$, consider an adversary $\mathcal{A}$ that asks tokens for l arbitrary keys $k_1, \ldots, k_l$ in the domain of G, and for l ciphertexts encrypting random message $s_1, \ldots, s_l$ from the key space of G. Now, in order to simulate the view of such an adversary, the simulator needs to produce tokens $\{\mathtt{Tok}_i\}$ and ciphertext $\{\mathsf{Ct}_j\}$ such that for every $i, j \in [\ell]$, $G_{s_j}(k_i) = \mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}_j, \mathtt{Tok}_i)$. Thus, the simulator needs to "encode" the function outputs, which correspond to l^2 pseudo-random bits, into $2l \cdot \lambda < l^2$ bits (l tokens and l ciphertexts of length λ). Since a pseudo-random string cannot be compressed by an efficient adversary, we get a contradiction.

Extending the technique. We note that a similar technique can be used to eliminate the use of a collision-resistant hash function in the impossibility of [BO12] and in the impossibility for non-black-box simulation given in the previous section. In both cases the hash function is used to bind the simulator to it previous answers by embedding a short hash of these answers in the next adaptive queries. Instead we can use many adaptive key queries (in the case of [BO12]) or many ciphertext queries (in the impossibility for non-adaptive queries) to encode the entire answer of the simulator (without a shrinking hash) in the adaptive queries.

In the case of adaptive queries, we get an impossibility for adversaries that makes an unbounded number of ciphertext queries and an unbounded number of adaptive key queries. This impossibility does not use random oracles and holds even against a computationally *unbounded* simulator.

In the case of non-adaptive queries, we get an impossibility for adversaries that makes an unbounded number of non-adaptive key queries and an unbounded number of ciphertext queries. This impossibility uses pseudo-random function but does not use collision-resistant hashing.

We finally remark that our negative result for fully non-adaptive adversaries and adaptive adversaries is in keeping with our standard model compiler that yields SIM-secure FE with unbounded number of adaptive key queries.

5.3 Functional Encryption for Functionalities with Long Output

Gorbunov, Vaikuntanathan and Wee [GVW12] give a positive result for bounded-query simulation-based secure FE. The main drawback of their result is that the ciphertext size depends on the size of the functionality. Recently, Goldwasser *et al.* [GKP+12] construct a simulation-based secure FE with "compact" ciphertexts for boolean functions. However, for functionalities with longer output, the ciphertexts in both of these constructions grows linearly with the output length of the functionality. Here, we show that this is, in fact, *inherent.* More specifically, we show that it is impossible to construct FE schemes where the ciphertext length is *independent* of the output length of the functionality.

We start by defining syntax for functional encryption where the ciphertext size is independent of the function output length. Existing definition of FE [GVW12,BSW11] considers functionalities with some fixed ratio between the input and output length. In particular, the length of ciphertexts in the resulting scheme may depend on this ratio. It is desirable to define FE such that at the time of encryption, the encrypting party does not need to place an upper bound on the size or output length of the functions that can be evaluated from the ciphertext.

Let $F : K_{n,t} \times M_n \to \Sigma_t$ be a function family where $K_{n,t}$ is the key space, M_n is the message space and Σ_t is the output space. (Note that the key space is parameterized according to both the input and output length).

A functional encryption scheme FE for F consists of four algorithms ($\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Eval}$) defined as follows. Let λ be the security parameter and $n = \mathsf{poly}(\lambda)$, $t = \mathsf{poly}(\lambda)$.

- $\mathsf{FE.Setup}(1^\lambda)$ is a PPT algorithm that takes as input (the unary representation of) the security parameter λ and outputs the public and the master secret keys ($\mathsf{Pk}, \mathsf{Msk}$).
- $\mathsf{FE.KeyGen}(\mathsf{Msk}, k)$ is a PPT algorithm that takes as input the master secret key Msk and a key $k \in K_{n,t}$, and outputs a token $\mathtt{Tok}_k$.
- $\mathsf{FE.Enc}(\mathsf{Pk}, m)$ is a PPT algorithm that takes as input the public key Pk and a plaintext $m \in M_n$, and outputs a ciphertext Ct.
- $\mathsf{FE.Eval}(\mathsf{Ct}, \mathtt{Tok})$ is a PPT algorithm that takes as input a ciphertext Ct and a token $\mathtt{Tok}$, and outputs $\sigma \in \Sigma_t$.

Correctness: We make the following *correctness* requirement from FE.
For all $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^n)$, all $k \in K_{n,t}$ and $m \in M_n$, for $\mathtt{Tok} \leftarrow \mathsf{FE.KeyGen}(\mathsf{Msk}, k)$ and $\mathsf{Ct} \leftarrow \mathsf{FE.Enc}(\mathsf{Pk}, m)$, we have that $\mathsf{FE.Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}) = F(k, m)$ except with negligible probability.

Impossibility of FE *with ciphertext size independent of function output length.* We now briefly sketch an impossibility result for constructing FE schemes where ciphertext size is independent of the function output length. We consider security against non-adaptive adversaries. In particular, we construct a non-adaptive adversary $\mathcal{A}$ who makes one (non-adaptive) token query and one ciphertext query such that the view of $\mathcal{A}$ cannot be simulated. (Our impossibility does not assume unbounded collusion as in [AGVW12].) For simplicity of presentation, we sketch an impossibility with respect to the block-box simulation-security definition of [AGVW12]. This can be extended to rule out also non-black box simulator by making use of techniques in [BO12] as used previously in this section.

Let $l - 1$ denote the a priori upper bound on the ciphertext length of an FE scheme. Let G be a pseudo-random generator that expands an n-bit seed to l bits. We will consider the functionality $F(k, s) = G(s)$. Now consider an adversary $\mathcal{A}$ who first makes a token query for a key k. On receiving the token $\mathtt{Tok}_k$, $\mathcal{A}$ makes a ciphertext query for ransom message s in the domain of G. Now, in order to simulate the view of such an adversary, the simulator first needs to produce a token $\mathtt{Tok}_k$, then the simulator is given the function output $G(s)$. The simulator is then required to produce a ciphertext Ct such that $G(s) = \mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_k)$. Thus, on the one hand, the simulator needs to "encode" the function output into Ct of length at most $l - 1$ bits, while on the other hand, the function output is an l-bit pseudo-random string. Since a pseudo-random string cannot be compressed by an efficient adversary, we get a contradiction.

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 657–677, 2015.

[AGG12] Joel Alwen, Rosario Gennaro, and Dov Gordon. On the relationship between functional encryption and fully homomorphic encryption. Manuscript, 2012.

[AGVW12] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. Cryptology ePrint Archive, Report 2012/468, 2012. `http://eprint.iacr.org/`.

[AKW16] Shashank Agrawal, Venkata Koppula, and Brent Waters. Impossibility of simulation secure functional encryption even with random oracles. Cryptology ePrint Archive, Report 2016/959, 2016. `http://eprint.iacr.org/`.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[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.

[BF12] Manuel Barbosa and Pooya Farshim. On the semantic security of functional encryption schemes. To appear at PKC, 2012.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 1–18, 2001.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. 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/`.

[Bon07] Dan Boneh. Bilinear groups of composite order (invited talk). In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *PAIRING 2007: 1st International Conference on Pairing-based Cryptography*, volume 4575 of *Lecture Notes in Computer Science*, page 1, Tokyo, Japan, July 2–4, 2007. Springer, Berlin, Germany.

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

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.

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

[BWY11] Mihir Bellare, Brent Waters, and Scott Yilek. Identity-based encryption secure against selective opening attack. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 235–252, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.

[CI16] Angelo De Caro and Vincenzo Iovino. On the power of rewinding simulators in functional encryption. *Designs, Codes and Cryptography*, pages 1–27, 2016. `http://dx.doi.org/10.1007/s10623-016-0272-x`.

[DCIP13] Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Fully secure hidden vector encryption. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, volume 7708 of *Lecture Notes in Computer Science*, pages 102–121. Springer, 2013.

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

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013.

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 480–511, 2016.

[GKP$^+$12] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Succinct functional encryption and applications: Reusable garbled circuits and beyond. (To appear at STOC 2013) Cryptology ePrint Archive, Report 2012/733, 2012. `http://eprint.iacr.org/`.

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

[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. Predicate encryption for circuits. To appear at STOC, 2013.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 163–172. ACM, 2015.

[IŻ15] Vincenzo Iovino and Karol Żebrowski. Simulation-based secure functional encryption in the random oracle model. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on*

Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings, pages 21–39, 2015.

[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.

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 318–335, Cambridge, UK, April 15–19, 2012. 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.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. 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.

[SW12] Amit Sahai and Brent Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2012/592, 2012. `http://eprint.iacr.org/`.

[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.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 678–697, 2015.

A Hidden Vector Encryption is not Preimage Sampleable

In this section we prove that if HVE is pre-image sampleable then we can decide 3SAT in polynomial time. We start by reviewing the notion of a *pre-image sampleable* functionality by O'Neill [O'N10].

Definition 19 [[O'N10]] Functionality $F : K \times M \to \{0,1\}$ is *pre-image sampleable* (PS, for short) if there exists a *sampler* algorithm Sam such that for all *PPT adversaries* $\mathcal{A}$,

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

for a negligible function ν. For $q > 0$, we say that F is *q-pre-image sampleable* (q-PS, for short) if the sampler algorithm Sam is guaranteed to work only with respect to adversaries $\mathcal{A}$ that output at most q pairs (k_i, b_i).

The following theorem is from [O'N10].

Theorem 20 If functionality F is PS then any IND-Secure functional encryption scheme FE for F is also SIM-Secure for non-adaptive adversaries.

Next we prove that the fact that the HVE functionality is PS has unexpected complexity-theoretic consequences.

Theorem 21 If HVE is PS then SAT can be decided in probabilistic polynomial time.

Proof. Let Sam be a sampler algorithm for HVE and consider the following algorithm $\mathcal{B}$ that, on input a Boolean formula $\Phi = \phi_1 \wedge \ldots \wedge \phi_c$ in CNF with c clauses and l variables $x_1, \ldots, x_l$, performs the following computation. In the description of $\mathcal{B}$, we will identify l-bit strings with truth assignment to variables $x_1, \ldots, x_l$.

For each clause ϕ_i of Φ, $\mathcal{B}$ computes key $k_i = (k_{1,i}, \ldots, k_{l,i}) \in \{0, 1, \star\}^l$ in the following way. Let $h \in \{0,1\}^l$ be a truth assignment that does not satisfy the i-th clause ϕ_i. For $j = 1, \ldots, l$, $\mathcal{B}$ sets $k_{j,i}$ in the following way

$$k_{j,i} = \begin{cases} h_j, & \text{if } x_j \text{ appears in } \ \phi_i; \\ 1 - h_j, & \text{if } \bar{x}_j \text{appears in } \ \phi_i; \\ \star, & \text{otherwise.} \end{cases}$$

Keys $k_1, \ldots, k_c$ enjoy the following (easy to verify) property. Let $m \in \{0,1\}^l$ be a truth assignment over the variables $x_1, \ldots, x_l$. Then, m satisfies Φ iff $\mathsf{HVE}(m, k_i) = 0$ for $i = 1, \ldots, c$.

$\mathcal{B}$ then runs algorithm Sam on input $((k_1, 0), \ldots, (k_c, 0))$ and let m be Sam's output. If m is a satisfying truth assignment for Φ then $\mathcal{B}$ decides that Φ is satisfiable. Otherwise, $\mathcal{B}$ decides that Φ is not satisfiable.

Let us now prove that $\mathcal{B}$'s output is correct with high probability. Notice that Sam is only guaranteed to work if it is given in input a sequence (k_i, b_i) for which there exists an $m \in \{0,1\}^l$ such that $b_i = \mathsf{HVE}(m, k_i)$. So we distinguish two cases. In the first case, we assume that Φ is satisfiable. Then the input of Sam is exactly as required by Definition 19 and thus, except with negligible probability, Sam outputs m such that $\mathsf{HVE}(m, k_i) = 0$ for $i = 1, \ldots, c$. By our previous observation such an m is a satisfying assignment for Φ and $\mathcal{B}$ is correct.

Suppose instead that Φ is not satisfiable. Then Sam will not output a satisfying assignment and $\mathcal{B}$ is correct.

B The Bilinear Setting and Complexity Assumptions

In this section we describe the bilinear setting with groups of composite order and the assumption that we will use to prove adaptive security of the scheme presented in Section 4.1.

Composite order bilinear groups were first used in Cryptography by [BGN05] (see also [Bon07]). We suppose the existence of an efficient group generator algorithm $\mathcal{G}$ which takes as input the security parameter λ and outputs a description $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ of a bilinear setting, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N, and $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map with the following properties:

1. (Bilinearity) $\forall\ g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$ it holds that $\mathbf{e}(g^a, h^b) = \mathbf{e}(g, h)^{ab}$.
2. (Non-degeneracy) $\exists\ g \in \mathbb{G}$ such that $\mathbf{e}(g, g)$ has order N in $\mathbb{G}_T$.

We assume that the group descriptions of $\mathbb{G}$ and $\mathbb{G}_T$ include generators of the respective cyclic groups. We require that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map $\mathbf{e}$ are computable in deterministic polynomial time in λ. In our construction we will make hardness assumptions for bilinear settings whose order N is product of five distinct primes each of length $\Theta(\lambda)$. For an integer m dividing N, we let $\mathbb{G}_m$ denote the subgroup of $\mathbb{G}$ of order m. From the fact that the group is cyclic, it is easy to verify that if g and h are group elements of co-prime orders then $\mathbf{e}(g, h) = 1$. This is called the *orthogonality* property and is a crucial tool in our constructions.

In our construction we will make the following hardness assumptions for bilinear settings whose order N is product of five distinct primes each of length $\Theta(\lambda)$.

Assumption 1. The assumption is a subgroup-decision type assumption for bilinear settings. More formally, we have the following definition. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4p_5, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick

$$A_3 \leftarrow \mathbb{G}_{p_3},\ A_4 \leftarrow \mathbb{G}_{p_4}, \mathcal{A}_{13} \leftarrow \mathbb{G}_{p_1p_3},\ A_{12} \leftarrow \mathbb{G}_{p_1p_2}, \quad T_0 \leftarrow \mathbb{G}_{p_1p_3},\ T_1 \leftarrow \mathbb{G}_{p_2p_3}\ ,$$

and set $D = (\mathcal{I}, A_3, A_4, A_{13}, A_{12})$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 1 to be

$$\mathsf{Adv}_1^{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|\ .$$

We say that Assumption 1 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_1^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 2. The assumption is a subgroup-decision type assumption for bilinear settings. More formally, we have the following definition. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4p_5, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick

$$A_1 \leftarrow \mathbb{G}_{p_1},\ A_2 \leftarrow \mathbb{G}_{p_2},\ A_3 \leftarrow \mathbb{G}_{p_3},\ A_4 \leftarrow \mathbb{G}_{p_4}, \quad T_0 \leftarrow \mathbb{G}_{p_3},\ T_1 \leftarrow \mathbb{G}_{p_3p_5}\ ,$$

and set $D = (\mathcal{I}, A_1, A_2, A_3, A_4)$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 2 to be

$$\mathsf{Adv}_2^{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|\ .$$

We say that Assumption 2 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_2^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 3. The assumption is a subgroup-decision type assumption for bilinear settings. More formally, we have the following definition. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4p_5, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick

$$A_1 \leftarrow \mathbb{G}_{p_1},\ A_2 \leftarrow \mathbb{G}_{p_2},\ A_3 \leftarrow \mathbb{G}_{p_3},\ A_4 \leftarrow \mathbb{G}_{p_4},\ A_{15} \leftarrow \mathbb{G}_{p_1p_5}, \quad T_0 \leftarrow \mathbb{G}_{p_1},\ T_1 \leftarrow \mathbb{G}_{p_1p_5}\ ,$$

and set $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_{15})$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 3 to be

$$\mathsf{Adv}_3^{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|\ .$$

We say that Assumption 3 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_3^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 4. The assumption is a subgroup-decision type assumption for bilinear settings. More formally, we have the following definition. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4p_5, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick

$$A_2 \leftarrow \mathbb{G}_{p_2},\ A_3 \leftarrow \mathbb{G}_{p_3},\ A_4 \leftarrow \mathbb{G}_{p_4},\ A_{14} \leftarrow \mathbb{G}_{p_1p_4},\ A_{15} \leftarrow \mathbb{G}_{p_1p_5}, \quad T_0 \leftarrow \mathbb{G}_{p_1p_4p_5},\ T_1 \leftarrow \mathbb{G}_{p_4p_5}\ ,$$

and set $D = (\mathcal{I}, A_2, A_3, A_4, A_{14}, A_{15})$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 4 to be

$$\mathsf{Adv}_4^{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|\ .$$

We say that Assumption 4 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_4^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 5 *(General Diffie-Hellman for composite order groups).* The assumption is a kind of Diffie-Hellman assumption in the bilinear setting of composite order. More formally, we have the following definition. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4p_5, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick

$$A_1 \leftarrow \mathbb{G}_{p_1},\ A_2 \leftarrow \mathbb{G}_{p_2},\ A_3 \leftarrow \mathbb{G}_{p_3},\ A_4, B_4, C_4, D_4 \leftarrow \mathbb{G}_{p_4},\ A_5 \leftarrow \mathbb{G}_{p_5},\ \alpha, \beta \leftarrow \mathbb{Z}_N\ ,$$

$$T_0 = A_1^{\alpha\beta} \cdot D_4,\ T_1 \leftarrow \mathbb{G}_{p_1p_4}\ ,$$

and set $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_5, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4)$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 5 to be

$$\mathsf{Adv}_5^{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|\ .$$

We say that Assumption 5 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_5^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

B.1 General Subgroup Decision Assumption

In this section we show that assumptions $1, 2, 3$ and 4 are special cases of the General Subgroup Decision Assumption introduced by Bellare *et al.* [BWY11] that is defined for bilinear groups of composite order product of m distinct primes, $p_1, \ldots, p_m$. We use $m = 5$.

For each non-empty subset $S \subseteq [m]$ we denote by $\mathbb{G}_S$ the subgroup of order $\prod_{i \in S} p_i$ in the bilinear group $\mathbb{G}$. Then the assumption is stated in the following way: Pick a random bilinear setting $\mathcal{I} = (N = p_1 \cdots p_m, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$. Let $S_0, S_1, S_2, \ldots, S_k$ be non-empty subsets of $[m]$ such that for each $2 \leq j \leq k$, either $S_j \cap S_0 = S_j \cap S_1 = \emptyset$ or $S_j \cap S_0 \neq \emptyset$ and $S_j \cap S_1 \neq \emptyset$. and then pick

$$Z_2 \leftarrow \mathbb{G}_{S_2}, \ldots, Z_k \leftarrow \mathbb{G}_{S_k},\ T_0 \leftarrow \mathbb{G}_{S_0},\ T_1 \leftarrow \mathbb{G}_{S_1}\ ,$$

and set $D = (\mathcal{I}, Z_2, \ldots, Z_k)$. We define the advantage of a PPT adversary $\mathcal{A}$ in breaking General Subgroup Decision Assumption to be

$$\mathsf{Adv}_{\mathsf{GSD}}^{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|\ .$$

We say that the General Subgroup Decision Assumption holds for generator $\mathcal{G}$ if, for all $S_0, \ldots, S_k$ that satisfy the conditions above and for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_{\mathsf{GSD}}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

We now show that Assumptions $1, 2, 3$ and 4 are a special case of the General Subgroup Decision Assumption. In all cases we let $m = 5$.

- For Assumption 1, we let $S_0 = \{1, 3\}, S_1 = \{2, 3\}, S_2 = \{3\}, S_3 = \{1, 3\}, S_4 = \{1, 2\}, S_5 = \{4\}$.
- For Assumption 2, we let $S_0 = \{3\}, S_1 = \{3, 5\}, S_2 = \{1\}, S_3 = \{2\}, S_4 = \{3\}, S_5 = \{4\}$.
- For Assumption 3, we let $S_0 = \{1\}, S_1 = \{1, 5\}, S_2 = \{1\}, S_3 = \{2\}, S_4 = \{3\}, S_5 = \{4\}, S_6 = \{1, 5\}$.
- For Assumption 4, we let $S_0 = \{1\}, S_1 = \{1, 5\}.S_2 = \{2\}, S_3 = \{3\}, S_4 = \{4\}, S_5 = \{1, 5\}$.

It is easy to see that in all cases the condition given in the assumption is verified.

C Proof of Theorem 16

C.1 Overview

Our simulator Sim consists of three algorithms: $\mathsf{Sim.Setup}$, $\mathsf{Sim.Enc}$, and $\mathsf{Sim.KeyGen}$ that share a common state. Algorithm $\mathsf{Sim.Setup}$ produces two public keys Pk and Pk'. Pk is given to the adversary (that may use it to generate his own ciphertexts) and Pk' instead is used to generate the simulated ciphertext. $\mathsf{Sim.Enc}$ is used to compute the simulated ciphertext and takes as input the public key Pk' and the sequence $(\boldsymbol{y}_k, z_k)_{k=1}^{q_1}$, where $z_k = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x})$, $\boldsymbol{x}$ is the plaintext output by the adversary and q_1 is the number of queries asked by the adversary in the first stage. $\mathsf{Sim.KeyGen}$ instead is used to answer the adaptive queries asked by the adversary after seeing the simulated ciphertext. When the adversary asks to see a token for $\boldsymbol{y}$, $\mathsf{Sim.KeyGen}$ is invoked with the master secret key Msk, the value $\boldsymbol{y}$ and the value $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is the challenge ciphertext output by the adversary. We stress that the simulator does not have access to $\boldsymbol{x}$.

Algorithm $\mathsf{Sim.Setup}$ constructs two public keys Pk (that is given to the adversary) and Pk' (that is used to produce the simulated ciphertext) and a master secret key Msk (that is used to answer the queries of the adversary). In public key Pk the $t_{i,b}$'s are encoded in the $\mathbb{G}_{p_2}$ part of the $T_{i,b}$'s (instead of the $\mathbb{G}_{p_1}$ part as in normal public key) whereas Pk' is generated correctly. Then, notice that the simulated ciphertext is independent from Pk, and this gives us enough freedom to manipulate the $\mathbb{G}_{p_1}$ part of the simulated ciphertext and tokens the adversary sees in the second stage. The subgroup $\mathbb{G}_{p_3}$ will help us to hide this transition. Algorithm $\mathsf{Sim.Enc}$ simulates the encryption of the challenge plaintext $\boldsymbol{x}$ provided by the adversary. It receives on input the queries $\boldsymbol{y}_k$, $k = 1, \ldots, q_1$, asked by the adversary in the first stage (the non-adaptive one) and the values $z_k = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x})$, for $k = 1, \ldots, q_1$.

$\mathsf{Sim.Enc}$ does not have access to $\boldsymbol{x}$ and by Theorem 21 the pre-image sampleability cannot be used to sample an $\boldsymbol{x}'$ such that $\mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}) = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}')$ for all k. But still some information about $\boldsymbol{x}$ can be derived from the inputs. Indeed, observe that if $\mathsf{HVE}(\boldsymbol{y}, \boldsymbol{x}) = 1$, then $\boldsymbol{y}$ and $\boldsymbol{x}$ coincide in all non-star entries of $\boldsymbol{y}$. For these positions, $\mathsf{Sim.Enc}$ computes the ciphertext Ct just like the encryption algorithm; for all remaining positions, the ciphertext output by $\mathsf{Sim.Enc}$ has a random $\mathbb{G}_{p_1}$ part.

Notice that such a simulated ciphertext Ct is then compatible with all the tokens the adversary sees in first stage in the sense that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_{\boldsymbol{y}_k}) = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x})$, for $k = 1, \ldots, q_1$. Unfortunately, the simulation is not adequate. In the second stage, the adversary could ask to see a token $\mathtt{Tok}_{\boldsymbol{y}}$ for a vector $\boldsymbol{y}$ such that $\mathsf{HVE}(\boldsymbol{y}, \boldsymbol{x}) = 1$ and we could have that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_{\boldsymbol{y}}) = 0$. This is because the $\mathbb{G}_{p_1}$ part of simulated ciphertext Ct and the token $\mathtt{Tok}_{\boldsymbol{y}}$ do not cancel out correctly upon decryption. Thus we remove the $\mathbb{G}_{p_1}$ part from the matching tokens the adversary sees in the second stage and use $\mathbb{G}_{p_5}$. Specifically, $\mathbb{G}_{p_5}$ will be introduced in the simulated ciphertext and matching second stage tokens in such a way that it will cancel out upon decryption and will provide us enough entropy to remove the $\mathbb{G}_{p_1}$ part from the adaptive tokens. Thus, each component of the ciphertext computed by $\mathsf{Sim.Enc}$ contains also a random $\mathbb{G}_{p_5}$ part that will be used for constructing the answers to the adaptive queries of the adversary. Notice the analogy between the $\mathbb{G}_{p_5}$ part and the flag used in the trapdoor circuits. If the $\mathbb{G}_{p_5}$ part is absent the ciphertext is in normal mode, otherwise it acts in trapdoor mode.

C.2 Full Proof

We start by describing the three algorithms used by Sim.

Algorithm $\mathsf{Sim.Setup}$: constructs two public keys Pk (that is given to the adversary) and Pk' (that is used to produce the simulated ciphertext) and a master secret key Msk (that is used to answer the queries of the adversary). In public key Pk the $t_{i,b}$'s are encoded in the $\mathbb{G}_{p_2}$ part of the $T_{i,b}$'s (instead of the $\mathbb{G}_{p_1}$ part as in normal public key) whereas Pk' is generated correctly. Then, notice that the simulated ciphertext is independent from Pk, and this gives us enough freedom to manipulate the $\mathbb{G}_{p_1}$ part of the simulated ciphertext and tokens the adversary sees in the second stage. The subgroup $\mathbb{G}_{p_3}$ will help us to hide this transition.

Specifically, $\mathsf{Sim.Setup}$ randomly chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4p_5, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization, and random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$. Then, for $i \in [\ell]$ and $b \in \{0,1\}$, $\mathsf{Sim.Setup}$ picks random $t_{i,b} \in Z_N$, $R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$ and $T'_{i,b} = g_1^{t_{i,b}} \cdot R'_{i,b}$. The public keys are $\mathsf{Pk} = [\mathcal{I}, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and $\mathsf{Pk}' = [\mathcal{I}, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$. $\mathsf{Sim.Setup}$ returns $(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk})$.

Algorithm $\mathsf{Sim.Enc}$: simulates the encryption of the challenge plaintext $\boldsymbol{x}$ provided by the adversary. It receives on input the queries $\boldsymbol{y}_k$, $k = 1, \ldots, q_1$, asked by the adversary in the first stage (the non-adaptive one) and the values $z_k = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x})$, for $k = 1, \ldots, q_1$.

$\mathsf{Sim.Enc}$ does not have access to $\boldsymbol{x}$ and by Theorem 21 the pre-image sampleability cannot be used to sample an $\boldsymbol{x}'$ such that $\mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}) = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}')$ for all k. But still some information about $\boldsymbol{x}$ can be derived from the inputs. Indeed, observe that if $\mathsf{HVE}(\boldsymbol{y}, \boldsymbol{x}) = 1$, then $\boldsymbol{y}$ and $\boldsymbol{x}$ coincide in all non-star entries of $\boldsymbol{y}$. For these positions, $\mathsf{Sim.Enc}$ computes the ciphertext Ct just like the encryption algorithm; for all remaining positions, the ciphertext output by $\mathsf{Sim.Enc}$ has a random $\mathbb{G}_{p_1}$ part.

Notice that such a simulated ciphertext Ct is then compatible with all the tokens the adversary sees in first stage in the sense that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_{\boldsymbol{y}_k}) = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x})$, for $k = 1, \ldots, q_1$. Unfortunately, the simulation is not adequate. In the second stage, the adversary could ask to see a token $\mathtt{Tok}_{\boldsymbol{y}}$ for a vector $\boldsymbol{y}$ such that $\mathsf{HVE}(\boldsymbol{y}, \boldsymbol{x}) = 1$ and we could have that $\mathsf{Eval}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{Tok}_{\boldsymbol{y}}) = 0$. This is because the $\mathbb{G}_{p_1}$ part of simulated ciphertext Ct and the token $\mathtt{Tok}_{\boldsymbol{y}}$ do not cancel out correctly upon decryption. Thus we remove the $\mathbb{G}_{p_1}$ part from the matching tokens the adversary sees in the second stage and use $\mathbb{G}_{p_5}$. Specifically, $\mathbb{G}_{p_5}$ will be introduced in the simulated ciphertext and matching second stage tokens in such a way that it will cancel out upon decryption and will provide us enough entropy to remove the $\mathbb{G}_{p_1}$ part from the adaptive tokens. Thus, each component of the ciphertext computed by $\mathsf{Sim.Enc}$ contains also a random $\mathbb{G}_{p_5}$ part that will be used for constructing the answers to the adaptive queries of the adversary.

More formally, for a sequence $Q = (\boldsymbol{y}_k, z_k)_{k=1}^{q_1}$ with $z_k = \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x})$, for $k = 1, \ldots, q_1$, we let MPos denote the set of indices $1 \le i \le \ell$ for which there exists $k \in \{1, \ldots, q_1\}$ such that $z_k = 1$ and $y_{k,i} \neq \star$. Notice that, by the observation above, for all $i \in \mathsf{MPos}$, the value x_i can be derived from the sequence Q. Then, $\mathsf{Sim.Enc}(\mathsf{Pk}, (\boldsymbol{y}_k, z_k)_{k=1}^{q_1})$ can be described as follows. The algorithm parses Pk as $\mathsf{Pk} = [\mathcal{I}, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and, for all $i \in \mathsf{MPos}$, derives the value x_i from the input sequence. Then, the algorithm chooses random $s \in \mathbb{Z}_N$ and, for each $i \in [\ell]$, random $Z_i \in \mathbb{G}_{p_3}$, and, for each $i \in [\ell]$: If $i \notin \mathsf{MPos}$, randomly select $r_i \in \mathbb{Z}_N$ and set $X_i = T_{i,0}^{r_i} \cdot g_5^{s_i} \cdot Z_i$. If $i \in \mathsf{MPos}$, set $X_i = T_{i,x_i}^{s} \cdot g_5^{s_i} \cdot Z_i$. $\mathsf{Sim.Enc}$ returns one output, the challenge ciphertext $\mathsf{Ct} = (X_i)_{i\in[\ell]}$, and keeps the vector $(s_i)_{i\in[\ell]}$ in the state so that it can be used by algorithm $\mathsf{Sim.KeyGen}$.

Algorithm $\mathsf{Sim.KeyGen}$: simulates the answer to the second stage queries of the adversary. It receives as input the master secret key Msk, the vector $\boldsymbol{y}$ for which the token has to be simulated and the value $z = \mathsf{HVE}(\boldsymbol{y}, \boldsymbol{x})$, where $\boldsymbol{x}$ is the challenge plaintext.

For each $j \in S_{\boldsymbol{y}}$, $\mathsf{Sim.KeyGen}$ selects random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\boldsymbol{y}}} a_j = 0$. Then, it distinguishes two cases.

$\boldsymbol{z} = 0$: In this case, the algorithm generates a token with a *random* $\mathbb{G}_{p_1}$ part. Specifically, for each $j \in S_{\boldsymbol{y}}$, $\mathsf{Sim.KeyGen}$ chooses random $c_j \in \mathbb{Z}_N$ and sets $Y_j = g_1^{c_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j$.

$\boldsymbol{z} = 1$: In this case, the algorithm generates a token *without* the $\mathbb{G}_{p_1}$ part and with a $\mathbb{G}_{p_5}$ part that will cancel out against the simulated ciphertext upon decryption. Specifically, for each $j \in S_{\boldsymbol{y}}$, $\mathsf{Sim.KeyGen}$ sets $Y_j = g_2^{a_j/t_{j,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j$.

We remind the reader that the vector $(s_j)_{j\in[\ell]}$ is the same vector used by $\mathsf{Sim.Enc}$ (and is stored in the state of the simulator). Also, note that the key output by $\mathsf{Sim.KeyGen}$ behaves as expected when matched against the ciphertext output by $\mathsf{Sim.Enc}$.

Thus proving Theorem Theorem 16 requires showing that the following two experiments are indistinguishable

```
$\mathsf{RealExp}^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}, \boldsymbol{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
Output: $(\mathsf{Pk}, \boldsymbol{x}, \alpha)$
```

```
$\mathsf{IdealExp}^{\mathcal{A}}_{\mathsf{Sim}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(\mathsf{Pk}', (\boldsymbol{y}_k, \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}))_{k=1}^{q_1})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot))}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
Output: $(\mathsf{Pk}, \boldsymbol{x}, \alpha)$
```

and we do so by using three intermediate hybrids that we call H_1, H_2 and H_3. The proof consists of four main steps (see Figure 2 for a quick reference).

Real	H_1	H_2	H_3	Ideal
Setup	Sim.Setup	Sim.Setup	Sim.Setup	Sim.Setup
Enc	Enc	Sim.A.Enc	Sim.A.Enc	Sim.Enc
KeyGen	KeyGen	KeyGen	Sim.KeyGen	Sim.KeyGen

Fig. 2. The hybrids used in the proof of security.

The first step of our proof consists in constructing the public key by means of $\mathsf{Sim.Setup}$. This has the effect of projecting the public key (and thus the ciphertexts the adversary constructs by himself) to a different subgroup from the one of the challenge ciphertext. Specifically:

```
$H_1^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Pk}', \boldsymbol{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
Output: $(\mathsf{Pk}, \boldsymbol{x}, \alpha)$
```

The first step of the proof consists in proving that the outputs of $\mathsf{RealExp}^{\mathcal{A}}$ and $H_1^{\mathcal{A}}$ are indistinguishable for all PPT adversaries $\mathcal{A}$.

The second step modifies the simulated ciphertext by adding a $\mathbb{G}_{p_5}$ part. This will be used in successive experiments to answer to the adversary's adaptive token queries. Specifically:

```
$H_2^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(\mathsf{Pk}', \boldsymbol{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
**Output:** $(\mathsf{Pk}, m, \alpha)$
```

where algorithm Sim.A.Enc proceeds as follows.

Algorithm Sim.A.Enc on input public key $\mathsf{Pk}' = [\mathcal{I}, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, the algorithm chooses random $s \in \mathbb{Z}_N$. Then, for $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ and random $s_i \in \mathbb{Z}_N$, sets

$$X_i = {T'_{i,x_i}}^s \cdot g_5^{s_i} \cdot Z_i \ ,$$

returns the tuple $(X_i)_{i\in[\ell]}$ and stores the vector $(s_i)_{i\in[\ell]}$ in the state so that it can be used by Sim.KeyGen.

The second step of the proof consists in proving that the outputs of $H_1^{\mathcal{A}}$ and $H_2^{\mathcal{A}}$ are indistinguishable for all PPT adversaries $\mathcal{A}$.

The third step computes the simulated answers using the Sim.KeyGen algorithm that adds a $\mathbb{G}_{p_5}$ part that cancels out with the one added in the simulated ciphertext.

```
$H_3^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(\mathsf{Pk}', \boldsymbol{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot))}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
**Output:** $(\mathsf{Pk}, \boldsymbol{x}, \alpha)$
```

Thus the third step of the proof consists in the defining hybrid experiment $H_3^{\mathcal{A}}$ and in proving that it is indistinguishable from $H_2^{\mathcal{A}}$ for all PPT adversaries $\mathcal{A}$.

The fourth step consists in proving that hybrid $H_3^{\mathcal{A}}$ is indistinguishable from the ideal experiment for all PPT adversaries $\mathcal{A}$.

This concludes the proof of Theorem 16. In the following we give the proof for each transition.

C.3 The first step of the proof

In this section we prove that Real and H_1 are indistinguishable.

Lemma 22 If Assumption 1 holds, then for all PPT adversaries $\mathcal{A}$, $\mathsf{Real}^{\mathcal{A}} \approx_c {H_1}^{\mathcal{A}}$.

Proof. Suppose there exists an adversary $\mathcal{A}$ for which $\mathsf{Real}^{\mathcal{A}}$ and ${H_1}^{\mathcal{A}}$ are distinguishable. Then, we show a PPT algorithm $\mathcal{B}$ which receives an instance of Assumption 1, $(\mathcal{I}, A_3, A_4, A_{13}, A_{12})$ with challenge T and, depending on the nature of T, simulates $\mathsf{Real}^{\mathcal{A}}$ or ${H_1}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$ sets $g_{12} = A_{12}, g_3 = A_3, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and sets

$$T_{i,b} = T^{t_{i,b}} \quad \text{and} \quad T'_{i,b} = A_{13}^{t_{i,b}} \ .$$

Then $\mathcal{B}$ sets

$$\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \quad \text{and} \quad \mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and

$$\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and starts the interaction with $\mathcal{A}$ on input Pk.

Token Queries: $\mathcal{B}$ uses Msk to answer the token queries of $\mathcal{A}$.

Ciphertext: $\mathcal{B}$ computes $\mathsf{Ct} = \mathsf{Enc}(\mathsf{Pk}', \boldsymbol{x})$.

This concludes the description of algorithm $\mathcal{B}$.

The view of $\mathcal{A}$ consists of Pk, Ct and the answer of the queries. The answers of the queries are distributed, independently from the nature of the challenge T, as in $\mathsf{Real}^{\mathcal{A}}$ and as in $H_1{}^{\mathcal{A}}$.

If $T \in \mathbb{G}_{p_1p_3}$ then Pk is constructed exactly as in $\mathsf{Real}^{\mathcal{A}}$. Moreover, the ciphertext Ct, even though is constructed using Pk', has the same distribution of a ciphertext constructed using Pk and thus the view of $\mathcal{A}$ is the same as in $\mathsf{Real}^{\mathcal{A}}$.

On the other hand, when $T \in \mathbb{G}_{p_2p_3}$ then Pk and Ct are distributed as in $H_1{}^{\mathcal{A}}$. □

C.4 The second step of the proof

In this section we prove that H_1 is indistinguishable from H_2.

Lemma 23 If Assumption 2 holds then, for all PPT adversaries $\mathcal{A}$, $H_1{}^{\mathcal{A}} \approx_c H_2{}^{\mathcal{A}}$.

Proof. Suppose there exists an adversary $\mathcal{A}$ for which $H_1{}^{\mathcal{A}}$ and $H_2{}^{\mathcal{A}}$ are not indistinguishable. Then, we show a PPT algorithm $\mathcal{B}$ which receives an instance of Assumption 2, $(\mathcal{I}, A_1, A_2, A_3, A_4)$ and challenge T and, depending on the nature of T, simulates $H_1{}^{\mathcal{A}}$ or $H_2{}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$ sets $g_2 = A_2$, $g_{12} = A_1 \cdot A_2, g_3 = A_3, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets

$$T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b} \quad \text{and} \quad T'_{i,b} = g_1^{t_{i,b}} \cdot R'_{i,b} \ .$$

Then $\mathcal{B}$ sets

$$\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \quad \text{and} \quad \mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and

$$\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and starts the interaction with $\mathcal{A}$ on input Pk.

Token Queries: $\mathcal{B}$ runs algorithm KeyGen on input Msk to answer token queries.

Simulated Ciphertext: $\mathcal{B}$ generates the simulated ciphertext for $\boldsymbol{x}$ as follows. $\mathcal{B}$ chooses, for $i \in [\ell]$, random $r_i \in \mathbb{Z}_N$ and random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = g_1^{t_{i,x_i}} \cdot T^{r_i} \cdot Z_i$$

This concludes the description of algorithm $\mathcal{B}$.

First of all, notice that, independently from the nature of challenge T, $\mathsf{Pk}, \mathsf{Pk}'$ and Msk are distributed as in the output of $\mathsf{Sim.Setup}$ and thus like in ${H_1}^{\mathcal{A}}$ and ${H_2}^{\mathcal{A}}$. Similarly for the answers to the token queries. Finally it is easy to see that if $T \in \mathbb{G}_{p_3}$ then Ct is distributed as in ${H_1}^{\mathcal{A}}$ and if $T \in \mathbb{G}_{p_3 p_5}$ then Ct is distributed as in ${H_2}^{\mathcal{A}}$. □

C.5 The third step of the proof

In this section we prove that H_2 is indistinguishable from H_3.

Lemma 24 If Assumptions 3, 4, and 5 hold then, for all PPT adversaries $\mathcal{A}$, $H_2^{\mathcal{A}} \approx_c H_3^{\mathcal{A}}$.

The proof of the lemma above consists of three substeps.

First Substep. In the *first substep*, we define a new experiment called $\mathsf{TypeAKeys}$ which differs from H_2 in the way the token queries of the second stage are answered. More specifically, the token for vector $\boldsymbol{y}$ such that $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 1$ contains a $\mathbb{G}_{p_5}$ part that is related with the one of the simulated ciphertext. The remaining tokens are like those in H_2. We will prove that, under Assumption 3, $H_2 \approx_c \mathsf{TypeAKeys}$.

Second Substep. In the *second substep*, we define a new experiment called $\mathsf{TypeBKeys}$ which differs from $\mathsf{TypeAKeys}$ in the way the token for matching queries of the second stage are generated. More specifically, if the adversary asks for the token for a vector $\boldsymbol{y}$ such that $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 1$, then the answer does not contain a $\mathbb{G}_{p_1}$ part. The tokens for $\boldsymbol{y}$ such that $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 0$ instead are computed as in $\mathsf{TypeAKeys}$. We will prove that, under Assumption 4, $\mathsf{TypeAKeys} \approx_c \mathsf{TypeBKeys}$.

Third Substep. Let q_2 be the number of token queries made by the adversary in the second stage. In the *third substep* we define, for $k = 0, \ldots, q_2$, a new experiment called $H_{3,k}$ which differs from $\mathsf{TypeBKeys}$ in the way the the second stage token queries are answered. More specifically, the first k tokens asked by the adversary are modified in the following way. The token for $\boldsymbol{y}$ such that $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 0$ contains a random $\mathbb{G}_{p_1}$ part. Instead if $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 1$ then the token is computed as in $\mathsf{TypeBKeys}$. The tokens for the remaining $q_2 - k$ queries are computed like in experiment $\mathsf{TypeBKeys}$. We observe that $\mathsf{TypeBKeys} = H_{3,0}$ and that $H_{3,q_2} = H_3$.

We will prove that, under Assumption 5, $H_{3,k-1} \approx_c H_{3,k}$.

The first substep Let us start by formally defining experiment $\mathsf{TypeAKeys}$ in terms of the algorithm $\mathsf{Sim.A.KeyGen}$ used to answer the token queries of the second stage.

Algorithm $\mathsf{Sim.A.KeyGen}$ receives in input Msk, $\boldsymbol{y}$ and $z = \mathsf{HVE}(\boldsymbol{y}, \boldsymbol{x})$ where $\boldsymbol{x}$ is the challenge plaintext provided by the adversary and distinguishes two cases.

$z = 0$: The answer to the query is computed by running KeyGen on input Msk and $\boldsymbol{y}$.

$z = 1$: For each $j \in S_{\boldsymbol{y}}$, the algorithm chooses random $W_j \in \mathbb{G}_{p_4}$, random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\boldsymbol{y}_i}} a_j = 0$, and sets

$$Y_j = g_{12}^{a_j / t_{j, y_j}} \cdot g_5^{a_j / s_j} \cdot W_j \ .$$

Notice that the vector $(s_j)_{j \in [\ell]}$ is the vector stored in the state by $\mathsf{Sim.A.Enc}$.

We define hybrid experiment $\mathsf{TypeAKeys}^{\mathcal{A}}$ as follows.

$$\begin{array}{l}
\mathsf{TypeAKeys}^{\mathcal{A}}(1^\lambda, 1^\ell)\\
(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell);\\
(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk});\\
\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(\mathsf{Pk}', \boldsymbol{x});\\
\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.A.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot))}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st});\\
\textbf{Output: } (\mathsf{Pk}, m, \alpha)
\end{array}$$

Lemma 25 If Assumption 3 holds then, for all PPT $\mathcal{A}$, $H_2^{\mathcal{A}} \approx_c \mathsf{TypeAKeys}^{\mathcal{A}}$.

Proof. Suppose there exists an adversary $\mathcal{A}$ for which $H_2{}^{\mathcal{A}}$ and $\mathsf{TypeAKeys}^{\mathcal{A}}$ are distinguishable. Then, we show a PPT algorithm $\mathcal{B}$ that receives an instance of Assumption 3, consisting of $(\mathcal{I}, A_1, A_2, A_3, A_4, A_{15})$ and challenge T, and, depending on the nature of T, simulates $H_2{}^{\mathcal{A}}$ or $\mathsf{TypeAKeys}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$ sets $g_2 = A_2$, $g_{12} = A_1 \cdot A_2, g_3 = A_3, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets

$$T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b} \quad \text{and} \quad T'_{i,b} = g_1^{t_{i,b}} \cdot R'_{i,b} \ .$$

Then $\mathcal{B}$ sets

$$\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \quad \text{and} \quad \mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and

$$\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and starts the interaction with $\mathcal{A}$ on input Pk.

First Stage Token Queries: $\mathcal{B}$ computes the token for $\boldsymbol{y}$ by running KeyGen on input Msk and $\boldsymbol{y}$.

Simulated Ciphertext: $\mathcal{B}$ generates the simulated ciphertext for vector $\boldsymbol{x}$ as follows. $\mathcal{B}$ chooses random $s \in \mathbb{Z}_N$ and, for $i \in [\ell]$, random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = A_{15}^{s \cdot t_{i,x_i}} \cdot Z_i$$

Second Stage Token Queries: $\mathcal{B}$ generates the token for $\boldsymbol{y}$ in the following way.

If $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 0$, the token is computed by running KeyGen on input Msk and $\boldsymbol{y}$.

If $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 1$, the token is computed as follows. For each $j \in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j\in S_{\boldsymbol{y}}} a_j = 0$. Then $\mathcal{B}$, for each $j \in S_{\boldsymbol{y}}$, sets:

$$Y_j = T^{a_j/t_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

This concludes the description of algorithm $\mathcal{B}$.

Now we observe that the output of $\mathcal{B}$'s setup is distributed like the output of algorithm $\mathsf{Sim.Setup}$ and thus like in H_2 and $\mathsf{TypeAKeys}$. Similarly, the simulated ciphertext is distributed like the output of $\mathsf{A.Enc}$ on input Pk' and $\boldsymbol{x}$ and thus exactly as in H_2 and $\mathsf{TypeAKeys}$. Finally, let us consider the answers to the second stage queries. If $T \in \mathbb{G}_{p_1}$ then the answer to the query for $\boldsymbol{y}$ has the same distribution as the output of KeyGen on input Msk and $\boldsymbol{y}$ and thus it is distributed as in H_2. On the other hand, if $T \in \mathbb{G}_{p_1p_5}$ then the answer to the query for $\boldsymbol{y}$ has the same distribution as the output of $\mathsf{Sim.A.KeyGen}$ on input $\mathsf{Msk}, \boldsymbol{y}$ and $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y})$ just like in $\mathsf{TypeAKeys}$. □

The second substep Let us start by formally defining experiment $\mathsf{TypeBKeys}$ in terms of the algorithm $\mathsf{Sim.B.KeyGen}$ that is used to answer the token queries of the second stage.

Algorithm $\mathsf{Sim.B.KeyGen}$ receives in input Msk, $\boldsymbol{y}$ and $z = \mathsf{HVE}(\boldsymbol{y}, \boldsymbol{x})$, where $\boldsymbol{x}$ is the challenge plaintext provided by the adversary and distinguishes two cases.

$z = 0$: The answer to the query is computed by running KeyGen on input Msk and $\boldsymbol{y}$.
$z = 1$: For each $j \in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j\in S_{\boldsymbol{y}}} a_j = 0$ and sets

$$Y_j = g_2^{a_j/t_{j,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \ .$$

Notice that vector $(s_j)_{j\in[\ell]}$ is the vector stored in the state by $\mathsf{Sim.A.Enc}$.

We define hybrid experiment $\mathsf{TypeBKeys}^{\mathcal{A}}$ as follows.

```
$\mathsf{TypeBKeys}^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(\mathsf{Pk}', \boldsymbol{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.B.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot))}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
**Output:** $(\mathsf{Pk}, m, \alpha)$
```

Lemma 26 If Assumption 4 holds then, for all PPT $\mathcal{A}$, $\mathsf{TypeAKeys}^{\mathcal{A}} \approx_c \mathsf{TypeBKeys}^{\mathcal{A}}$.

Proof. Suppose there exists an adversary $\mathcal{A}$ for which $\mathsf{TypeAKeys}^{\mathcal{A}}$ and $\mathsf{TypeBKeys}^{\mathcal{A}}$ are distinguishable. Then, we show a PPT algorithm $\mathcal{B}$ that receives an instance of Assumption 4, consisting of $(\mathcal{I}, A_2, A_3, A_4, A_{14}, A_{15})$ and challenge T, and, depending on the nature of T, simulates $\mathsf{TypeAKeys}^{\mathcal{A}}$ or $\mathsf{TypeBKeys}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$ sets $g_2 = A_2$, $g_3 = A_3, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets

$$T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b} \quad \text{and} \quad T'_{i,b} = A_{15}^{t_{i,b}} \cdot R'_{i,b} \ .$$

Then $\mathcal{B}$ sets

$$\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \quad \text{and} \quad \mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and

$$\mathsf{Msk} = [\perp, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$

and starts the interaction with $\mathcal{A}$ on input Pk. Notice that $\mathcal{B}$ is unable to compute a complete Msk as it does not have access to an element from $\mathbb{G}_{p_1p_2}$.

First Stage Token Queries: $\mathcal{B}$ generates the token for $\boldsymbol{y}$ in the following way. For each $i \in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_i \in \mathbb{G}_{p_4}$, random $a_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_{\boldsymbol{y}}} a_i = 0$ and then sets

$$Y_i = A_{14}^{a_i/t_{i,y_i}} \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i \ .$$

Simulated Ciphertext: $\mathcal{B}$ generates the simulated ciphertext for vector $\boldsymbol{x}$ as follows. $\mathcal{B}$ chooses random $s \in \mathbb{Z}_N$ and, for $i \in [\ell]$, random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = A_{15}^{s\cdot t_{i,x_i}} \cdot Z_i$$

Second Stage Token Queries: On input $\mathsf{Msk}, \boldsymbol{y}$ and $z = \mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y})$, $\mathcal{B}$ generates the token for vector $\boldsymbol{y}$ in the following way.

For each $j \in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_y} a_j = 0$. Then $\mathcal{B}$ distinguishes two cases.

$z = 0$**:** For each $j \in S_{\boldsymbol{y}}$, $\mathcal{B}$ sets

$$Y_j = A_{14}^{a_i/t_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

$z = 1$**:** For each $j \in S_{\boldsymbol{y}}$, $\mathcal{B}$ sets

$$Y_j = T^{a_j/t_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

This concludes the description of algorithm $\mathcal{B}$.

We start by observing that Pk has the same distribution of the public key constructed by algorithm $\mathsf{Sim.Setup}$ just like in $\mathsf{TypeAKeys}$ and $\mathsf{TypeBKeys}$. Similarly, the answers to the token queries of the first stage are distributed like the output of KeyGen, despite the fact that $\mathcal{B}$ has an incomplete Msk. Similarly, despite the fact that the public key Pk' computed by $\mathcal{B}$ differs from the one of $\mathsf{Sim.Setup}$, it is not difficult to see that Ct has the same distribution of the output of $\mathsf{Sim.A.Enc}$ on input a public key Pk' output by $\mathsf{Sim.Setup}$. Let us now look at the answers to the token queries of the second stage. It is easy to see that queries for $\boldsymbol{y}$ such that $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 0$ are distributed as the output of KeyGen. Suppose that $T \in \mathbb{G}_{p_1 p_4 p_5}$. Then the answers to matching queries are distributed like in the output of algorithm $\mathsf{Sim.A.KeyGen}$ and thus like in $\mathsf{TypeAKeys}$. On the other hand, if $T \in \mathbb{G}_{p_4 p_5}$ then the answers to matching queries are distributed like in the output of algorithm $\mathsf{Sim.B.KeyGen}$ and thus like in $\mathsf{TypeBKeys}$.

□

The third substep In this section we prove that, for any PPT adversary $\mathcal{A}$, $\mathsf{TypeBKeys}^{\mathcal{A}}$ and $H_3^{\mathcal{A}}$ are indistinguishable, under Assumption 5.

For any adversary $\mathcal{A}$ that makes q_2 queries in the second stage, we define a sequence of hybrid experiments $H_{3,k}^{\mathcal{A}}$, for $k = 0, \ldots, q_2$, such that

$$H_{3,0}^{\mathcal{A}} = \mathsf{TypeBKeys}^{\mathcal{A}} \quad \text{and} \quad H_{3,q_2}^{\mathcal{A}} = H_3^{\mathcal{A}} \ .$$

Then if $\mathsf{TypeBKeys}^{\mathcal{A}}$ and $H_3^{\mathcal{A}}$ can be distinguished then there must exist $k \in \{1, \ldots, q_2\}$ such that hybrid experiments $H_{3,k-1}^{\mathcal{A}}$ and $H_{3,k}^{\mathcal{A}}$ are indistinguishable. Moreover, from the definition of experiment $H_{3,k}$, it is clear that if k-th token query of the second stage is a matching query (that is, $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 1$) then $H_{3,k-1}^{\mathcal{A}}$ and $H_{3,k}^{\mathcal{A}}$ coincide. Therefore, if $\mathcal{A}$ is such that $\mathsf{TypeBKeys}^{\mathcal{A}}$ and $H_3^{\mathcal{A}}$ can be distinguished there must exist k such that $H_{3,k-1}^{\mathcal{A}}$ and $H_{3,k}^{\mathcal{A}}$ can be distinguished and the k-th token query of $\mathcal{A}$ in the second stage is with non-negligible probability a non-matching query (that is, $\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 0$).

Let us now define experiment $H_{3,k}$. In $H_{3,k}$ second stage token queries are answered by running a parametrized version of algorithm $\mathsf{Sim.KeyGen}$ that, with a slight abuse of notation, we also call $\mathsf{Sim.KeyGen}$.

Algorithm $\mathsf{Sim.KeyGen}$ receives as input the master secret key Msk, the vector $\boldsymbol{y}$ for which the token must be computed and the value $z = \mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y})$. In addition, $\mathsf{Sim.KeyGen}$ receives the number i of the query and the value k.

For each $j \in S_{\boldsymbol{y}}$, $\mathsf{Sim.KeyGen}$ selects random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\boldsymbol{y}}} a_j = 0$. Then, it distinguishes two cases.

$z = 0$: The depending on the query index i, $\mathcal{B}$ does the following.

– If $i \leq k$: For each $j \in S_{\boldsymbol{y}}$, $\mathsf{Sim.KeyGen}$ chooses random $c_j \in \mathbb{Z}_N$ and sets

$$Y_j = g_1^{c_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

– If $i > k$: For each $j \in S_{\boldsymbol{y}}$, $\mathsf{Sim.KeyGen}$ sets

$$Y_j = g_{12}^{a_j/t_{j,y_j}} \cdot W_j \ .$$

$z = 1$: For each $j \in S_{\boldsymbol{y}}$, $\mathsf{Sim.KeyGen}$ sets

$$Y_j = g_2^{a_j/t_{j,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \ .$$

We remind the reader that the vector $(s_j)_{j\in[\ell]}$ is the vector stored in the state of the simulator.

We observe that for algorithm $\mathsf{Sim.B.KeyGen}$ is the parametrized version of $\mathsf{Sim.KeyGen}$ with $k = 0$. On the other hand, algorithm $\mathsf{Sim.KeyGen}$ is the parametrized version of $\mathsf{Sim.KeyGen}$ with $k = q_2$.

Next we define hybrid experiment $H_{3,k}^{\mathcal{A}}$, for an adversary $\mathcal{A}$ that ask q_2 token queries in the second stage and for $k = 0, \ldots, q_2$,

$H_{3,k}^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(\mathsf{Pk}', \boldsymbol{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot),\cdot,k)}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
Output: $(\mathsf{Pk}, \boldsymbol{x}, \alpha)$

Notice that $H_{3,0} = \mathsf{TypeBKeys}$ and $H_{3,q_2} = H_3$.

Lemma 27 If Assumption 5 holds then, for any PPT adversary $\mathcal{A}$, $\mathsf{TypeBKeys}^{\mathcal{A}} \approx_c H_3^{\mathcal{A}}$

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{TypeBKeys}$ and H_3 are distinguishable. We show a PPT algorithm $\mathcal{B}$ that receives an instance of Assumption 5, consisting of $(\mathcal{I}, A_1, A_2, A_3, A_4, A_5, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4)$ and challenge T, and, depending on the nature of T, simulates $H_{3,k-1}^{\mathcal{A}}$ or $H_{3,k}^{\mathcal{A}}$ with some non-negligible probability for a random $k \in \{1, \ldots, q_2\}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$ randomly chooses position $\mathsf{j} \in [\ell]$ and $b_{\mathsf{j}} \in \{0, 1\}$.
$\mathcal{B}$ sets $g_1 = A_1$, $g_2 = A_2$, $g_{12} = A_1 \cdot A_2$, $g_3 = A_3$, $g_4 = A_4$, $g_5 = A_5$ and, for each $i \in [\ell]$ and $b \in \{0, 1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Moreover, for all $(i, b) \in [\ell] \times \{0, 1\}$ with $(i, b) \neq (\mathsf{j}, b_{\mathsf{j}})$, $\mathcal{B}$ sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R'_{i,b}$.
Finally, $\mathcal{B}$ sets

$$\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \quad \text{and} \quad \mathsf{Pk}' = [N, g_3, (T'_{i,b})_{(i,b)\in([\ell]\times\{0,1\})\setminus\{(\mathsf{j},b_{\mathsf{j}})\}}]$$

and

$$\mathsf{Msk} = [g_{12}, g_4, , (t_{i,b})_{(i,b)\in([\ell]\times\{0,1\})\setminus\{(\mathsf{j},b_{\mathsf{j}})\}}]$$

and starts the interaction with $\mathcal{A}$ on input Pk.

First Stage Token Queries: In the first stage, $\mathcal{B}$ generates token for $\boldsymbol{y}$ in the following way. For each $i \in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_i \in \mathbb{G}_{p_4}$, random $a_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i \in S_{\boldsymbol{y}}} a_i = 0$ and then sets

$$Y_i = g_{12}^{a_i/t_{i,y_i}} \cdot W_i \ .$$

However, if $y_{\mathsf{j}} = b_{\mathsf{j}}$ then $\mathcal{B}$ sets

$$Y_{\mathsf{j}} = (A_1^{\beta} \cdot C_4)^{a_i} \cdot g_2^{a_{\mathsf{j}}/t_{\mathsf{j},y_{\mathsf{j}}}} \cdot W_{\mathsf{j}} \ .$$

This last setting has the effect of implicitly setting $t_{\mathsf{j},y_{\mathsf{j}}} \equiv 1/\beta \pmod{p_1}$.

Challenge Ciphertext: $\mathcal{B}$ generate the challenge ciphertext for vector $\boldsymbol{x}$ in the following way. If $x_{\mathsf{j}} = b_{\mathsf{j}}$ then $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ chooses random $s \in \mathbb{Z}_N$ and, for $i \in [\ell]$, random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = g_1^{s \cdot t_{i,x_i}} \cdot g_5^{s \cdot t_{i,x_i}} \cdot Z_i.$$

Second Stage Token Queries: At the start of the second stage of token queries, $\mathcal{B}$ picks a random $k \in \{1, \ldots, q_2\}$.

The i-th token query of the second stage for vector $\boldsymbol{y}$ is answered by $\mathcal{B}$ in the following way by distinguishing the following two cases.

$\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}) = 0$: We distinguish between the following three cases

$i < k$**:** For each $j \in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$, $c_j \in \mathbb{Z}_N$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\boldsymbol{y}}} a_j = 0$ and sets

$$Y_j = g_1^{c_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

$i = k$**:** If $y_{\mathsf{j}} \neq b_{\mathsf{j}}$ then $\mathcal{B}$ aborts.

Otherwise let $h \in [\ell]$ be such that $h \neq \mathsf{j}$ and $y_h \neq \star$ and set $S = S_{\boldsymbol{y}} \setminus \{\mathsf{j}, h\}$, Notice that such an h always exists since we assumed that each query contains at least two non-$\star$ entries.

Then, for each $j \in S$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ and sets

$$Y_j = g_{12}^{a_j/t_{j,y_j}} \cdot W_j \ .$$

Then, for position j, $\mathcal{B}$ chooses random $W_{\mathsf{j}} \in \mathbb{G}_{p_4}$ and random $a_{\mathsf{j}} \in \mathbb{Z}_N$ and sets

$$Y_{\mathsf{j}} = T \cdot g_2^{a_{\mathsf{j}}/t_{\mathsf{j},y_{\mathsf{j}}}} \cdot W_{\mathsf{j}}.$$

Finally, for position h, $\mathcal{B}$ sets

$$Y_h = (A_1^{\alpha} B_4)^{-1/t_{h,y_h}} \cdot g_1^{-s/t_{h,y_h}} \cdot g_2^{-(s+a_{\mathsf{j}})/t_{h,y_h}} \cdot W_h,$$

where $s = \sum_{j \in S} a_j$.

$i > k$**:** For each $j \in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\boldsymbol{y}}} a_j = 0$ and sets

$$Y_j = g_{12}^{a_j/t_{j,y_j}} \cdot W_j \ .$$

In the special case that $y_{\mathsf{j}} = b_{\mathsf{j}}$, then $\mathcal{B}$ sets

$$Y_{\mathsf{j}} = (A_1^{\beta} \cdot C_4)^{a_{\mathsf{j}}} \cdot g_2^{a_{\mathsf{j}}/t_{\mathsf{j},y_{\mathsf{j}}}} \cdot W_{\mathsf{j}} \ .$$

$\mathsf{HVE}(\boldsymbol{x},\boldsymbol{y}_i)=1$: For each $j\in S_{\boldsymbol{y}}$, $\mathcal{B}$ chooses random $W_j\in\mathbb{G}_{p_4}$ and $a_j\in\mathbb{Z}_N$ under the constraint that $\sum_{j\in S_{\boldsymbol{y}}} a_j=0$ and sets

$$Y_j=g_2^{a_j/t_{j,y_j}}\cdot g_5^{a_j/t_{j,y_j}}\cdot W_j.$$

This concludes the description of algorithm $\mathcal{B}$.

Let us now prove that the probability that $\mathcal{B}$ does not abort while interacting with $\mathcal{A}$ is non-negligible. First observe that the probability that $\mathcal{B}$ aborts while constructing the simulated ciphertext is 1/2. Indeed, the view of $\mathcal{A}$ up to this point is independent from j and b_{j} and thus, since b_{j} is chosen at random by $\mathcal{B}$, the probability that $x_{\mathsf{j}}=b_{\mathsf{j}}$ is 1/2. Let us now look at the probability that $\mathcal{B}$ aborts while answering the k-th token query of the second stage of $\mathcal{A}$, given that it has not aborted in the construction of the simulated ciphertext. In this case the view of $\mathcal{A}$ is independent from j and also remember that the probability that the k-th query of $\mathcal{A}$ is non-matching is non-negligible. If this is the case then there must exist one position j such that $y_j\neq\star$ and $y_j\neq x_j$. If $j=\mathsf{j}$ (which happens with non-negligible probability since $\mathcal{A}$'s view is independent from the value of j) then $\mathcal{B}$ does not abort. Indeed, in this case we would have $y_{\mathsf{j}}\neq x_j\neq b_j$ which implies that $y_{\mathsf{j}}=b_j$.

Let us now look at the view of $\mathcal{A}$ while interacting with $\mathcal{B}$. We observe that Pk has the same distribution of the corresponding output of $\mathsf{Sim.Setup}$. Even though $\mathcal{B}$ does not have a complete Msk as it misses $t_{\mathsf{j},b_{\mathsf{j}}}$, the answer of the first stage queries are distributed as the output of algorithm KeyGen on input Msk in which $t_{\mathsf{j},b_{\mathsf{j}}}\equiv 1/\beta \pmod{p_1}$. Given that $\mathcal{B}$ does not abort, it is straightforward to see that Ct constructed by $\mathcal{B}$ has the same distribution as the output of $\mathsf{Ct}\leftarrow\mathsf{Sim.A.Enc}$ on input Pk' and $\boldsymbol{x}$. Let us now look at the answers of the second stage queries. The matching queries and the first $k-1$ non-matching queries have the same distribution of the output of algorithms $\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot),\cdot,k)$ and $\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot),\cdot,k-1)$. Notice that, for both types of queries, the missing element of Msk plays no role. Similarly, the last q_2-k non-matching queries of the second stage are distributed as the output $\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot),\cdot,k)$ and $\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot),\cdot,k-1)$ in which the missing element of Msk is set equal to $t_{\mathsf{j},b_{\mathsf{j}}}\equiv 1/\beta \pmod{p_1}$. Let us finally look at the answer to the k-th query. If $T=A_1^{\alpha\beta}\cdot D_4$ then the answer of the query is distributed according to the output of $\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot),\cdot,k-1)$. Notice that if B does not abort the missing element of Msk plays no role. On the other hand, if T is random $\mathbb{G}_{p_1p_4}$ then the answer of the query is distributed according $\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot),\cdot,k)$. □

C.6 The fourth step of the proof

In this section we prove that H_3 is indistinguishable from $\mathsf{IdealExp}$.

Lemma 28 If Assumption 5 holds then, for all PPT adversaries $\mathcal{A}$, $H_3^{\mathcal{A}}\approx_c\mathsf{IdealExp}^{\mathcal{A}}$.

We start by defining $\ell+1$ intermediate hybrid experiments $I_1,\ldots,I_{\ell+1}$ such that $I_1=H_3$ and $I_{\ell+1}=\mathsf{IdealExp}$ and show that, for $f=1,\ldots,\ell+1$, I_f and I_{f+1} are indistinguishable, under Assumption 5.

To define I_f, we introduce a parametrized version of algorithm $\mathsf{Sim.Enc}$ that, with a slight abuse of notation, we also call $\mathsf{Sim.Enc}$.

The parametrized version of $\mathsf{Sim.Enc}$ takes as input public key $\mathsf{Pk}'=[N,g_3,(T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, the challenge ciphertext $\boldsymbol{x}$, the sequence $(\boldsymbol{y}_k,z_k)_{k=1}^{q_1}$ of the q_1 queries asked by the adversary in the first stage along with $z_k=\mathsf{HVE}(\boldsymbol{y}_k,\boldsymbol{x})$. In addition we let $\mathsf{Sim.Enc}$ take parameter $0\leq f\leq\ell$.

$\mathsf{Sim.Enc}$ returns a simulated ciphertext in which the $\mathbb{G}_{p_1}$ part of positions $i \leq f$ that do not belong to MPos is random. The remaining positions are well formed.

More formally, $\mathsf{Sim.Enc}$ chooses random $s \in \mathbb{Z}_N$, and, for each $i \in [\ell]$, random $Z_i \in \mathbb{G}_{p_3}$ and random $s_i \in \mathbb{Z}_N$. Then for each $i \in [\ell]$, $\mathsf{Sim.Enc}$ distinguishes the following cases.

– if $i < f$ and $i \notin \mathsf{MPos}$, $\mathsf{Sim.Enc}$ randomly selects $r_i \in \mathbb{Z}_N$ and sets

$$X_i = T'^{r_i}_{i,0} \cdot g_5^{s_i} \cdot Z_i.$$

– if $i < f$ and $i \in \mathsf{MPos}$, $\mathsf{Sim.Enc}$ sets

$$X_i = T'^{s}_{i,x_i} \cdot g_5^{s_i} \cdot Z_i.$$

– if $i \geq f$ $\mathsf{Sim.Enc}$ sets

$$X_i = T'^{s}_{i,x_i} \cdot g_5^{s_i} \cdot Z_i.$$

$\mathsf{Sim.Enc}$ returns the simulated ciphertext $\mathsf{Ct} = (X_i)_{i\in[\ell]}$ and stores the vector $(s_i)_{i\in[\ell]}$ in the state. Notice that if $f = \ell + 1$ the input $\boldsymbol{x}$ is not used by the algorithm and we obtain algorithm $\mathsf{Sim.Enc}$. On the other hand, if $f = 1$, we obtain algorithm $\mathsf{Sim.A.Enc}$.

Next, we define, for $1 \leq f \leq \ell + 1$, experiment $I_f^{\mathcal{A}}$ as follows.

```
$I_f^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(\mathsf{Pk}', \boldsymbol{x}, (\boldsymbol{y}_k, \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}))_{k=1}^{q_1}, f)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot))}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
**Output:** $(\mathsf{Pk}, \boldsymbol{x}, \alpha)$
```

Clearly, for all PPT adversaries $\mathcal{A}$, $I_1^{\mathcal{A}} = H_3$ and $I_{\ell+1}^{\mathcal{A}} = \mathsf{IdealExp}^{\mathcal{A}}$. Therefore to prove Lemma 28, it is enough to prove the following lemma.

Lemma 29 If Assumption 5 holds, then for all PPT adversaries $\mathcal{A}$, and for $f = 1, \ldots, \ell$, $I_f^{\mathcal{A}} \approx_c I_{f+1}^{\mathcal{A}}$.

To prove the above lemma, we introduce another sequence of intermediate games and make the following observation.

Observation 30 *If* $f \in \mathsf{MPos}$*, the output distributions of* $\mathsf{Sim.Enc}(\mathsf{Pk}, \boldsymbol{x}, (\boldsymbol{y}_k, \mathit{HVE}(\boldsymbol{y}_k, \boldsymbol{x})_{k=1}^{q}, f)$ *and* $\mathsf{Sim.Enc}(\mathsf{Pk}, \boldsymbol{x}, (\boldsymbol{y}_k, \mathit{HVE}(\boldsymbol{y}_k, \boldsymbol{x})_{k=1}^{q}, f + 1)$ *coincide.*

Therefore if $I_f^{\mathcal{A}}$ and $I_{f+1}^{\mathcal{A}}$ are distinguishable then it must be the case that $\mathcal{A}$ has a non-negligible probability of outputting a challenge plaintext $\boldsymbol{x}$ such that $f \notin \mathsf{MPos}$. For an adversary $\mathcal{A}$ that makes q_1 first stage token queries we introduce $2 \cdot (q+1)$ intermediate hybrid experiments $L_{f,0}^{\mathcal{A}}, \ldots, L_{f,q_1}^{\mathcal{A}}$ and $M_{f,q_1}^{\mathcal{A}}, \ldots, M_{f,0}^{\mathcal{A}}$ which differ in the way in which first stage token queries are answered. Specifically, first stage queries are answered by running the following algorithm.

Algorithm $\mathsf{Sim.C.KeyGen}$ takes as input the master secret key Msk, the query $\boldsymbol{y}$ for which a token has to be computed, the number $1 \le i \le q_1$ of the query, integer $1 \le f \le \ell + 1$ and integer $0 \le k \le q_1$. The algorithm distinguishes the following cases.

- $i \le k$ and $y_f \neq \star$:
  The $\mathbb{G}_{p_1}$ part of the token is random.
  For each $j \in S_{\boldsymbol{y}}$, the algorithm chooses random $W_j \in \mathbb{G}_{p_4}$, random $c_j \in \mathbb{Z}_N$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_y} a_j = 0$ and sets
  $$Y_j = g_1^{c_j} \cdot g_2^{a_j / t_{j,y_j}} \cdot W_j.$$
- $i \le k$ and $y_f = \star$: the algorithm returns the output of $\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$.
- $i > k$: the algorithm returns the output of $\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$.

We are now ready to describe experiments $L^{\mathcal{A}}_{f,k}$ and $M^{\mathcal{A}}_{f,k}$ for $f = 1, \ldots, \ell + 1$ and $k = 0, \ldots, q_1$.

```
$L^{\mathcal{A}}_{f,k}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{Sim.C.KeyGen}(\mathsf{Msk},\cdot,\cdot,f,k)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(\mathsf{Pk}', \boldsymbol{x}, (\boldsymbol{y}_k, \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}))_{k=1}^q, f-1)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot))}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
**Output:** $(\mathsf{Pk}, m, \alpha)$

$M^{\mathcal{A}}_{f,k}(1^\lambda, 1^\ell)$
$(\mathsf{Pk}, \mathsf{Pk}', \mathsf{Msk}) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{Sim.C.KeyGen}(\mathsf{Msk},\cdot,\cdot,f,k)}(\mathsf{Pk})$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(\mathsf{Pk}', \boldsymbol{x}, (\boldsymbol{y}_k, \mathsf{HVE}(\boldsymbol{y}_k, \boldsymbol{x}))_{k=1}^q, f)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(\mathsf{Msk},\cdot,\mathsf{HVE}(\boldsymbol{x},\cdot))}(\mathsf{Pk}, \mathsf{Ct}, \mathtt{st})$;
**Output:** $(\mathsf{Pk}, m, \alpha)$
```

Before continuing, we observe the following:

Observation 31 *For all PPT adversaries $\mathcal{A}$, $I^{\mathcal{A}}_f = L_{f,0}$.*
Directly from the definition of the experiments.

Observation 32 *For all PPT adversaries $\mathcal{A}$, $L^{\mathcal{A}}_{f,q_1} = M^{\mathcal{A}}_{f,q_1}$ for $f = 1, \ldots, \ell$, where q_1 is the number of first stage queries made by $\mathcal{A}$.*
From the definitions of the two experiments, it is clear that all the token queries are answered in the same way in both the experiments and all components X_i for $i \neq f$ of the challenge ciphertext are computed in the same way. Let us now look at X_f and more precisely to its $\mathbb{G}_{p_1}$ part. In L_{f,q_1}, the $\mathbb{G}_{p_1}$ part of X_f is computed as T'^{s}_{f,x_f} which is exactly how it is computed in M_{f,q_1} when $f \in \mathsf{MPos}$. On the other hand, when $f \notin \mathsf{MPos}$, the $\mathbb{G}_{p_1}$ part of X_f is chosen at random. However, observe that exponents $t_{f,0} \mod p_1$ and $t_{f,1} \mod p_1$ have not appeared in the answers to key queries since every query has either a $\star$ in position f (in which case position f of the answer is empty) or a non-$\star$ value in position f (in which case the $\mathbb{G}_{p_1}$ part of the element in position f of the answer is random). Therefore, we can conclude that the $\mathbb{G}_{p_1}$ part of the component X_f of the answer to the challenge query is also random in $\mathbb{G}_{p_1}$.

Observation 33 *For all PPT adversaries $\mathcal{A}$ and for $f = 1, \ldots, \ell - 1$, $M^{\mathcal{A}}_{f,0} = L^{\mathcal{A}}_{f+1,0}$.*
Indeed, in both experiments all key queries are answered correctly, and the challenge query in $M_{f,0}$ is by definition answered in the same way as in $L_{f+1,0}$.

By the above observations, for proving Lemma 29 it suffices to prove that $L_{f,k-1}$ and $L_{f,k}$ are indistinguishable and that $M_{f,k-1}$ and $M_{f,k}$ are indistinguishable, for $k = 1, \ldots, q_1$. In the next section we prove that, under Assumption 5, $L_{f,k-1}$ and $L_{f,k}$ are indistinguishable. The proof that $M_{f,k-1}$ and $M_{f,k}$ are indistinguishable is similar and omitted.

Indistinguishability of $L_{f,k}$ and $L_{f,k-1}$ We start by describing an algorithm $\mathcal{B}$ that takes as input f and k and an instance of Assumption 5 and interacts with an adversary $\mathcal{A}$. Then, provided that $\mathcal{A}$ outputs a challenge such that $f \notin \mathsf{MPos}$, $\mathcal{B}$ simulates with some non-negligible probability $L_{f,k}$ or $L_{f,k-1}$ depending on the nature of the challenge. This suffices to prove that the two hybrids are indistinguishable.

Description of algorithm $\mathcal{B}$

Input: Integers $1 \le f \le \ell + 1$ and $0 \le k \le q$, and a randomly chosen instance of Assumption 5 consisting of $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_5, A_1^{\alpha} B_4, A_1^{\beta} C_4)$ and challenge T which is either $T = A_1^{\alpha\beta} D_4$ or random $\mathbb{G}_{p_1 p_4}$.

Setup: $\mathcal{B}$ starts by constructing public parameters Pk. $\mathcal{B}$ sets $g_1 = A_1, g_2 = A_2, g_{12} = A_1 \cdot A_2$, $g_3 = A_3, g_4 = A_4, g_5 = A_5$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{B}$ sets

$$\mathsf{Pk} = [N, g_3, (T_{i,b})_{i \in [\ell], b \in \{0,1\}}] ,$$

and starts the interaction with $\mathcal{A}$ on input Pk.

Now, $\mathcal{B}$ guesses a position $\mathsf{j} \in [\ell]$ and a bit $b_\mathsf{j} \in \{0,1\}$ and chooses the following randomness that will be used in the $\mathbb{G}_{p_1}$ subgroup. Specifically, $\mathcal{B}$ chooses, for each $i \in [\ell]$ and $i \neq \mathsf{j}$, and $b \in \{0,1\}$, random $t'_{i,j} \in \mathbb{Z}_N$. Moreover, $\mathcal{B}$ chooses random $t'_{\mathsf{j},c_\mathsf{j}} \in \mathbb{Z}_N$ where $c_\mathsf{j} = 1 - b_\mathsf{j}$. Notice that the value $t'_{\mathsf{j},b_\mathsf{j}}$ is unknown to $\mathcal{B}$. It will be provided by the assumption as β at the exponent.

First Stage Secret Tokens: To simulate the output of $\mathsf{Sim.C.KeyGen}(\mathsf{Sk}, \boldsymbol{y}_i, f, k)$, $\mathcal{B}$ does the following way:

$i \le k$: We have the following mutually exclusive cases.

Case A.1: $y_f \neq \star$. In this case, $\mathcal{B}$ outputs a key whose $\mathbb{G}_{p_1}$ part is random.

Specifically, for each $j \in S_{\boldsymbol{y}_i}$, $\mathcal{B}$ chooses random a_j such that $\sum_{j \in S_{\boldsymbol{y}_j}} a_i = 0$, random $r_j \in \mathbb{Z}_N$, and random $W_j \in \mathbb{G}_{p_4}$. Then, for each $j \in S_{\boldsymbol{y}_i}$, $\mathcal{B}$ sets

$$Y_j = g_1^{r_j} \cdot g_2^{a_j / v_{i,y_j}} \cdot W_j .$$

Case A.2: $y_f = \star$. In this case, $\mathcal{B}$ outputs a key with a well-formed $\mathbb{G}_{p_1}$ part.

Specifically, $\mathcal{B}$, for each $j \in S_{\boldsymbol{y}_i}$, chooses random $W_j \in \mathbb{G}_{p_4}$, random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\boldsymbol{y}_i}} a_j = 0$ and then sets

$$Y_j = g_1^{a_j / t'_{j,y_j}} \cdot g_2^{a_j / t_{j,y_j}} \cdot W_j .$$

In the special case that $y_\mathsf{j} = b_\mathsf{j}$, then $\mathcal{B}$ sets

$$Y_\mathsf{j} = (A_1^{\beta} \cdot C_4)^{a_i} \cdot g_2^{a_\mathsf{j} / t_{\mathsf{j},y_\mathsf{j}}} \cdot W_\mathsf{j} .$$

$i = k$: We distinguish between the following cases:

Case B.1: $y_f = \star$. $\mathcal{B}$ performs the same steps of Case A.2.

Case B.2: $y_f \neq \star$ **and** $y_\mathsf{j} \neq b_\mathsf{j}$. In this case, $\mathcal{B}$ aborts.

Case B.3: $y_f \neq \star$ **and** $y_{\mathsf{j}} = b_{\mathsf{j}}$. $\mathcal{B}$ mounts T, the challenge of the assumption, in position j.

Specifically, let $S = S_{\boldsymbol{y}_i} \setminus \{\mathsf{j}, h\}$, where h is an index such that $y_{k,h} \neq \star$. Such an index h always exists since we assumed that each query contains at least two non-$\star$ entries. Then, for each $j \in S$, $\mathcal{B}$ sets

$$Y_j = g_{12}^{a_j/t'_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

Then, for position j, $\mathcal{B}$ sets

$$Y_{\mathsf{j}} = T \cdot g_2^{a_{\mathsf{j}}/t_{\mathsf{j},y_{\mathsf{j}}}} \cdot W_{\mathsf{j}} \ ,$$

and for position h, $\mathcal{B}$ sets

$$Y_h = (A_1^{\alpha} B_4)^{-1/t'_{h,y_h}} \cdot g_1^{-s/t'_{h,y_h}} \cdot g_2^{-(s+a_j)/t_{h,y_h}} \cdot W_h,$$

where $s = \sum_{j \in S} a_j$.

$i > k$: $\mathcal{B}$ handles these queries as in Case A.2, independently from whether $y_f = \star$ or $y_f \neq \star$.

Challenge Ciphertext: $\mathcal{B}$ receives $\boldsymbol{x}$ and has to compute a ciphertext. We distinguishes the following two cases:

Case C.1: $x_{\mathsf{j}} = b_{\mathsf{j}}$. In this case, $\mathcal{B}$ aborts.

Case C.2: $x_{\mathsf{j}} \neq b_{\mathsf{j}}$. In this case, $\mathcal{B}$ computes sequence $(\boldsymbol{y}_i, \mathsf{HVE}(\boldsymbol{y}_i, \boldsymbol{x}))_{i=1}^{q_1}$ for all queries $\boldsymbol{y}_i$ that it has received from $\mathcal{A}$ to induce the set MPos.

Then, $\mathcal{B}$ chooses random $s \in \mathbb{Z}_N$, and, for each $i \in [\ell]$, random $Z_i \in \mathbb{G}_{p_3}$ and random $s_i \in \mathbb{Z}_N$. Then, for each $i \in [\ell]$, if $i < f$ and $i \notin \mathsf{MPos}$, the algorithm randomly select $r_i \in \mathbb{Z}_N$ and sets

$$X_i = g_1^{r_i} \cdot g_5^{s_i} \cdot Z_i \ ,$$

otherwise, the algorithm sets

$$X_i = g_1^{s \cdot t'_{i,x_i}} \cdot g_5^{s_i} \cdot Z_i \ .$$

Second Stage Secret Tokens: To simulate the output of $\mathsf{Sim.KeyGen}(\mathsf{Msk}, \boldsymbol{y}_i, \mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}_i))$, $\mathcal{B}$ does the following way:

$\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}_i) = 0$: $\mathcal{B}$ generates a secret key with a random $\mathbb{G}_{p_1}$ part and without the $\mathbb{G}_{p_5}$ part.

Specifically, for each $j \in S_{\boldsymbol{y}_i}$, $\mathcal{B}$ chooses random a_j such that $\sum_{j \in S_{\boldsymbol{y}_j}} a_i = 0$, random $r_j \in \mathbb{Z}_N$, and random $W_j \in \mathbb{G}_{p_4}$. Then, for each $j \in S_{\boldsymbol{y}_i}$, $\mathcal{B}$ sets

$$Y_j = g_1^{r_j} \cdot g_2^{a_j/t_{i,y_j}} \cdot W_j \ .$$

$\mathsf{HVE}(\boldsymbol{x}, \boldsymbol{y}_i) = 1$: $\mathcal{B}$ generates a secret key without the $\mathbb{G}_{p_1}$ part and with a $\mathbb{G}_{p_5}$ part correlated with that of the ciphertext.

Specifically, for each $j \in S_{\boldsymbol{y}_i}$, $\mathcal{B}$ chooses random a_j such that $\sum_{j \in S_{\boldsymbol{y}_j}} a_i = 0$, and random $W_j \in \mathbb{G}_{p_4}$. Then, for each $j \in S_{\boldsymbol{y}_i}$, $\mathcal{B}$ sets

$$Y_j = g_2^{a_j/t_{i,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \ .$$

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

The algorithm $\mathcal{B}$ will be used to prove properties of experiments L. We can modify $\mathcal{B}$ so that, on input f and k, the challenge ciphertext is constructed by randomizing the $\mathbb{G}_{p_1}$ part also of the f-th component. The so modified algorithm, that we call $\mathcal{B}_2$, closely simulates the work of experiments M and will be used to prove properties of these experiments.

Let us define the following two events.

$\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k)$: denotes the event that $\mathcal{B}$ does not abort while computing the answer to the k-th query in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that $y_{k,f} = \star$ or $y_{k,\mathsf{j}} = b_{\mathsf{j}}$.

$\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{B}}(f,k)$: denotes the event that $\mathcal{B}$ does not abort while computing the ciphertext in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that adversary $\mathcal{A}$ outputs vector $\boldsymbol{x}$ such that $x_{\mathsf{j}} = c_{\mathsf{j}} = 1 - b_{\mathsf{j}}$.

We can modify, experiments H_1, $L(f,k)$ and $M(f,k)$ so that j and b_{j} are chosen just like $\mathcal{B}$ does. This modification makes the definitions of events $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{Exp}}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{Exp}}$ meaningful also for these experiments. We write $\mathsf{NotAbort}^{\mathcal{A}}_{2}$ as a shorthand for $\mathsf{NotAbort}^{\mathcal{A}}_{2,H_1}$.

Lemma 34 For all f, k and $\mathcal{A}$, $\Pr[\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k)] \geq \frac{1}{\ell}$.

Proof. The probability of $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k)$ is at least the probability that $y_{k,\mathsf{j}} = b_{\mathsf{j}}$. Moreover, the view of $\mathcal{A}$ up to the k-th key query is independent from b_{j} and j. Now observe that the $\boldsymbol{y}_k$ has at least two non-star entry and, provided that j is one of these (which happens with probability at least $2/\ell$), the probability that $y_{k,\mathsf{j}} = b_{\mathsf{j}}$ is $1/2$. □

Lemma 35 For all f, k and $\mathcal{A}$, $\Pr[\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{Exp}}(f,k)] \geq \frac{1}{2\ell}$ for $\mathsf{Exp} = \mathsf{L(f,k)}$.

Proof. $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{Exp}}(f,k)$ is the event that $y_{\boldsymbol{k},j} \neq x_{\mathsf{j}}$ in the experiment Exp. It is easy to see that the probability that j and b_{j} are correctly guessed such that $x_{\mathsf{j}} = c_{\mathsf{j}} = 1 - b_{\mathsf{j}}$ is at least $1/(2\ell)$, independently from the view of $\mathcal{A}$. □

Lemma 36 Suppose event $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k)$ occurs. If $T = T_1$ then $\mathcal{A}$'s view up to the challenge ciphertext in the interaction with $\mathcal{B}$ running on input (f,k) is the same as in $L_{f,k-1}$. If instead $T = T_2$ then $\mathcal{A}$'s view up to the challenge ciphertext in the interaction with $\mathcal{B}$ running on input (f,k) is the same as in $L_{f,k}$.

Moreover, suppose events $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k)$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{B}}(f,k)$ occur. If $T = T_1$ then $\mathcal{A}$'s total view in the interaction with $\mathcal{B}$ running on input (f,k) is the same as in $L_{f,k-1}$. If instead $T = T_2$ then $\mathcal{A}$'s total view in the interaction with $\mathcal{B}$ running on input (f,k) is the same as in $L_{f,k}$.

Proof. First observe that Pk has the same distribution as the public parameters seen by $\mathcal{A}$ in both experiments. The same holds for the answers to the first $(k-1)$ key queries and to the last $(q_1 - k)$. Let us now focus on the answer to the k-th key query. We have two cases:

Case 1: $y_f = \star$**.** Then the view of $\mathcal{A}$ in the interaction with $\mathcal{B}$ is independent from T (see Case B.1) and, on the other hand, by definition, the two experiments coincide. Therefore the lemma holds in this case.

Case 2: $y_f \neq \star$. Suppose $T = T_1 = A_1^{\alpha\beta} \cdot D_4$ and that $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k)$ occurs. Therefore, $y_{\mathsf{j}} = b_{\mathsf{j}}$ and $\mathcal{B}$'s answer to the k-th key query has the same distributions as in $L(f, k-1)$. On the other hand if T is random in $\mathbb{G}_{p_1 p_4}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k)$ occurs, the $\mathbb{G}_{p_1}$ parts of the Y_j's are random and thus the answer to the k-th query of $\mathcal{A}$ is distributed as in $L_{f,k}$.

For the second part of the lemma, we observe that the challenge ciphertext has the same distribution in both experiments and that, if $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{B}}(f,k)$ occurs, $\mathcal{B}$ properly constructs the challenge ciphertext. □

Let us now analyze the probability that $\mathcal{A}$ does output a challenge such that $f \notin \mathsf{MPos}$, this is crucial for $\mathcal{B}$ to successfully simulate with some non-negligible probability $L(f,k)$ or $L(f,k-1)$.

We start by introducing some notation.

$E^{\mathcal{A}}_{f,\mathsf{Exp}}$ is defined as the event that in experiment Exp the adversary $\mathcal{A}$ declare a challenge vector such that $f \notin \mathsf{MPos}$. When the adversary $\mathcal{A}$ is clear from the context we will simply write $E_{f,\mathsf{Exp}}$.

$E^{\mathcal{A}}_{f}$: is defined as the event that in experiment H_1, the adversary $\mathcal{A}$ declares a challenge vector such that $f \notin \mathsf{MPos}$. When the adversary $\mathcal{A}$ is clear from the context we will simply write E_f.

$E^{\mathcal{A}}_{f,\mathcal{B}}(f',k)$: we extend the definition of $E_{f,\mathsf{Exp}}$ to include the experiment played by $\mathcal{A}$ against the algorithm $\mathcal{B}$. Thus we denote by the event that in the interaction between $\mathcal{A}$ and $\mathcal{B}$ on input f' and k, $\mathcal{B}$ does not abort and $\mathcal{A}$ declares a challenge vector such that $f \notin \mathsf{MPos}$. If $\mathcal{A}, f'$ and k are clear from the context, we will simply write $E_{f,\mathcal{B}}$.

$E^{\mathcal{A}}_{f,f'}$: is defined as the event that during the execution of $I_{f'}$ adversary $\mathcal{A}$ outputs a challenge vector such that $f \notin \mathsf{MPos}$.

Observation 37 *For all PPT adversaries $\mathcal{A}$ and distinguisher $\mathcal{D}$ and all $1 \leq f \leq \ell$, we have that* $\Pr[\mathcal{D}(I(f)^{\mathcal{A}}) = 1 | \neg E_{f,f}] = \Pr[\mathcal{D}(I(f+1)^{\mathcal{A}}) = 1 | \neg E_{f,f+1}]$.

Proof. By definition of I, if the challenge vector is such that $f \in \mathsf{MPos}$, then $\mathcal{A}$'s view in I_f and I_{f+1} is the same. □

Observation 38 *For all PPT adversaries $\mathcal{A}$ and all $1 \leq f \leq \ell$, we have that* $\Pr[E^{\mathcal{A}}_{f,f}] = \Pr[E^{\mathcal{A}}_{f,f+1}]$.

Proof. The view of $\mathcal{A}$ in I_f up to the challenge ciphertext is independent from f. □

Therefore we can set $\Pr[E^{\mathcal{A}}_{f,f}] = \Pr[E^{\mathcal{A}}_{f,1}] = \Pr[E^{\mathcal{A}}_{f}]$.

Lemma 39 If Assumption 2 holds, then for $k = 1, \ldots, q$ and $f = 1, \ldots, \ell$, and for all PPT adversaries $\mathcal{A}$, $\left|\Pr[E^{\mathcal{A}}_{f,G}] - \Pr[E^{\mathcal{A}}_{f,H}]\right|$ and $\left|\Pr[\mathsf{NotAbort}^{\mathcal{A}}_{2,G}] - \Pr[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}]\right|$ are negligible functions of λ, for experiments $G = L(f, k-1)$ and $H = L(f,k)$.

Proof. We prove the lemma for $E_{f,G}$ and $E_{f,H}$. A similar reasoning holds for $\mathsf{NotAbort}^{\mathcal{A}}_{2,G}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,H}$. For the sake of contradiction, suppose that $\Pr[E^{\mathcal{A}}_{f,G}] \geq \Pr[E^{\mathcal{A}}_{f,H}] + \epsilon$ for some non-negligible ϵ. Then we can modify algorithm $\mathcal{B}$ into algorithm $\mathcal{B}$ with a non-negligible advantage in breaking Assumption 2. Algorithm $\mathcal{B}$ simply execute $\mathcal{B}$'s code. By Lemma 34 event $\mathsf{NotAbort}_{1,\mathcal{B}}$ occurs with probability at least $1/\ell$ and in this case $\mathcal{B}$ can continue the execution of $\mathcal{B}$'s code and receive the challenge vector from $\mathcal{A}$. At this point, $\mathcal{B}$ checks whether $f \notin \mathsf{MPos}$. If it is the case, $\mathcal{B}$ outputs 1; else $\mathcal{B}$ outputs 0. It is easy to see that, by Lemma 36, the above algorithm has a non-negligible advantage in breaking Assumption 2. □

The proof of the following corollary is straightforward from Lemma 39 and Observations 32-33.

Corollary 40 For all $f = 1, \dots, \ell + 1$ and $k = 0, \dots, q$, and all PPT adversaries $\mathcal{A}$, we have that, for $H = L_{f,k}$ $\left|\Pr[E^{\mathcal{A}}_{f,H}] - \Pr[E^{\mathcal{A}}_{f}]\right|$ and $\left|\Pr[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}] - \Pr[\mathsf{NotAbort}^{\mathcal{A}}_{2}]\right|$ are negligible.

We are now ready to prove that $L_{f,k-1} \approx_c L_{f,k}$. To do this let us define the event $\mathsf{Succ}^{\mathcal{A}}(f,k)$ as

$$\mathsf{Succ}^{\mathcal{A}}(f,k) := \mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{B}}(f,k) \wedge \ \mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{B}}(f,k) \wedge E^{\mathcal{A}}_{f,\mathcal{B}}(f,k). \tag{1}$$

When $\mathcal{A}$ is clear from the context we use the shortcut $\mathsf{Succ}(f,k)$.

We are now ready to prove Lemma 41.

Lemma 41 Suppose there exists an adversary $\mathcal{A}$, a distinguisher $\mathcal{D}$ and integers $1 \leq f \leq \ell$ and $1 \leq k \leq q$ such that $\left|\Pr[\mathcal{D}(G^{\mathcal{A}}) = 1] - \Pr[\mathcal{D}(H^{\mathcal{A}}) = 1]\right| \geq \epsilon$, where $G = L(f, k-1)$, $H = L(f,k)$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathsf{Adv}^{\mathcal{B}}_2 \geq \Pr[E_f] \cdot \epsilon/(2 \cdot \ell^2) - \nu(\lambda)$, for a negligible function ν.

Proof. Assume without loss of generality that $\Pr[\mathcal{D}(G^{\mathcal{A}}) = 1] \geq \Pr[\mathcal{D}(H^{\mathcal{A}}) = 1] + \epsilon$ and consider the following algorithm $\mathcal{B}$. $\mathcal{B}$ uses algorithm $\mathcal{B}$ as a subroutine and interacts with $\mathcal{A}$ on input integers f and k for which the above inequality holds, and an instance (D,T) of Assumption 2. If event $\mathsf{Succ}(f,k)$ does not occur, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ outputs $\mathcal{D}$'s output. Therefore we have

$$\begin{aligned}\Pr[\mathcal{B} \text{ outputs } 1 | T = T_1] = \Pr[\mathcal{B} \text{ outputs } 1 | T = T_1 \wedge \mathsf{Succ}(f,k)] \cdot \\ \Pr[\mathsf{Succ}(f,k) | T = T_1]\end{aligned} \tag{2}$$

By definition of $\mathsf{Succ}(f,k)$ we have

$$\begin{aligned}\Pr[\mathsf{Succ}(f,k)|T = T_1] &= \Pr[E_{f,\mathcal{B}} \wedge \mathsf{NotAbort}_{1,\mathcal{B}} \wedge \mathsf{NotAbort}_{2,\mathcal{B}} | T = T_1] \\ &= \Pr[\mathsf{NotAbort}_{1,\mathcal{B}} | T = T_1] \cdot \\ &\quad \Pr[E_{f,\mathcal{B}} \wedge \mathsf{NotAbort}_{2,\mathcal{B}} | \mathsf{NotAbort}_{1,\mathcal{B}} \wedge T = T_1].\end{aligned}$$

Now observe that event $\mathsf{NotAbort}_{1,\mathcal{B}}$ is determined before $\mathcal{B}$ uses T and thus

$$\Pr[\mathsf{NotAbort}_{1,\mathcal{B}} | T = T_1] = \Pr[\mathsf{NotAbort}_{1,\mathcal{B}}].$$

Moreover, by Lemma 36, if event $\mathsf{NotAbort}_{1,\mathcal{B}}$ occurs and $T = T_1$, the view of $\mathcal{A}$ up to Challenge Query is equal to the view of $\mathcal{A}$ in experiment G and thus

$$\Pr[E_{f,\mathcal{B}} \wedge \mathsf{NotAbort}_{2,\mathcal{B}} | \mathsf{NotAbort}_{1,\mathcal{B}} \wedge T = T_1] = \Pr[E_{f,G} \wedge \mathsf{NotAbort}_{2,G}]$$

whence

$$\begin{aligned}\Pr[\mathsf{Succ}^{\mathcal{A}}(f,k) | T = T_1] &= \Pr[\mathsf{NotAbort}_{1,\mathcal{B}}] \cdot \Pr[\mathsf{NotAbort}_{2,G} \wedge E_{f,G}] \\ &= \Pr[\mathsf{NotAbort}_{1,\mathcal{B}}] \cdot \Pr[\mathsf{NotAbort}_{2,G}] \cdot \Pr[E_{f,G}]\ ,\end{aligned}$$

where $\mathsf{NotAbort}_{2,G}$ and $E_{f,G}$ are independent. Finally, if $T = T_1$ and $\mathsf{Succ}^{\mathcal{A}}(f,k)$ occurs, then, by Lemma 36, $\mathcal{A}$'s view is exactly as in experiment G, and thus the probability that $\mathcal{B}$ outputs 1 is equal to the probability that $\mathcal{D}$ output 1. We can thus rewrite Eq. 2 as

$$\Pr[\mathcal{B}\text{ outputs }1|T = T_1] = \Pr[\mathcal{D}(G^{\mathcal{A}}) = 1]\cdot \\ \Pr[\mathsf{NotAbort}_{1,\mathcal{B}}]\cdot\Pr[\mathsf{NotAbort}_{2,G}]\cdot\Pr[E_{f,G}]$$

A similar reasoning yields

$$\Pr[\mathcal{B}\text{ outputs }1|T = T_2] = \Pr[\mathcal{D}(H^{\mathcal{A}}) = 1]\cdot \\ \Pr[\mathsf{NotAbort}_{1,\mathcal{B}}]\cdot\Pr[\mathsf{NotAbort}_{2,H}]\cdot\Pr[E_{f,H}]$$

By using Corollary 40, Lemma 34 and Lemma 35, we can conclude that there exists a negligible function ν such that we have

$$\begin{aligned}\mathsf{Adv}_2^{\mathcal{B}} &= \Pr[\mathsf{NotAbort}_{1,\mathcal{B}}]\cdot\Pr[\mathsf{NotAbort}_2]\cdot\Pr[E_f]\cdot \\ &\quad \big(\Pr[\mathcal{D}(G^{\mathcal{A}}) = 1] - \Pr[\mathcal{D}(H^{\mathcal{A}}) = 1]\big) - \nu(\lambda) \\ &\geq \frac{\epsilon}{2\ell^2}\cdot\Pr[E_f] - \nu(\lambda)\end{aligned}$$

□

The following Lemma can be proved by referring to algorithm $\mathcal{B}_2$. We omit further details since the proof is essentially the same as the one of Lemma 41.

Lemma 42 Suppose there exists an adversary $\mathcal{A}$, a distinguisher $\mathcal{D}$ and integers $1 \leq f \leq \ell + 1$ and $1 \leq k \leq q$ such that $\big|\Pr[\mathcal{D}(G^{\mathcal{A}}) = 1] - \Pr[\mathcal{D}(H^{\mathcal{A}}) = 1]\big| \geq \epsilon$, where $G = M_{f,k-1}$, $H = M_{f,k}$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathsf{Adv}_2^{\mathcal{B}} \geq \Pr[E_f]\cdot\epsilon/(2\cdot\ell^2) - \nu(\lambda)$, for a negligible function ν.

We are finally ready to prove Lemma 29.

Lemma 29. If Assumption 2 holds, $I_f \approx_c I_{f+1}$.

PROOF. Suppose that for some adversary $\mathcal{A}$, distinguisher $\mathcal{D}$ and $f \in [\ell]$

$$\big|\Pr[\mathcal{D}(I_f^{\mathcal{A}}) = 1] - \Pr[\mathcal{D}(I_{f+1}^{\mathcal{A}}) = 1]\big| \geq \epsilon\ . \tag{3}$$

Now recall that $I_f = L_{f,0}$ and $I_{f+1} = M_{f,0}$. Thus, there exists $1 \leq k \leq q$ such that:

$$\big|\Pr[\mathcal{D}(G^{\mathcal{A}}) = 1] - \Pr[\mathcal{D}(H^{\mathcal{A}}) = 1]\big| \geq \epsilon/(2q)\ ,$$

where $G = L_{f,k}$ and $H = L_{f,k-1}$ or where $G = M_{f,k}$ and $H = M_{f,k-1}$. Then by Lemma 41, in the former case, and by Lemma 42 in the latter, we can construct an adversary $\mathcal{B}$ against Assumption 2, such that

$$\mathsf{Adv}_2^{\mathcal{B}} \geq \frac{\epsilon}{4q\ell^2}\cdot\Pr[E_f] - \nu(\lambda)$$

Now it remains to estimate $\Pr[E_f]$. Notice that we can write

$$\Pr[\mathcal{D}(I_f^{\mathcal{A}}) = 1] = \Pr[E_{f,f}]\cdot\Pr[\mathcal{D}(I_f^{\mathcal{A}}) = 1|E_{f,f}]+ \\ \Pr[\neg E_{f,f}]\cdot\Pr[\mathcal{D}(I_f^{\mathcal{A}}) = 1|\neg E_{f,f}]\ ,$$

and

$$\begin{aligned}\Pr[\mathcal{D}(I^{\mathcal{A}}_{f+1}) = 1] = \Pr[E_{f,f+1}] \cdot \Pr[\mathcal{D}(I^{\mathcal{A}}_{f+1}) = 1 | E_{f,f+1}] +\\ \Pr[\neg E_{f,f+1}] \cdot \Pr[\mathcal{D}(I^{\mathcal{A}}_{f}) = 1 | \neg E_{f,f+1}]\end{aligned} ,$$

and by combining Equation 3 and Observations 37 and 38, we obtain

$$\Pr[E_f] \cdot \left|\Pr[\mathcal{D}(I^{\mathcal{A}}_{f}) = 1 | E_{f,f}] - \Pr[\mathcal{D}(I^{\mathcal{A}}_{f+1}) = 1 | E_{f,f+1}]\right| \geq \epsilon.$$

Thus, we can conclude that

$$\Pr[E_f] \geq \epsilon \ ,$$

and thus $\mathcal{B}$ as advantage

$$\mathsf{Adv}^{\mathcal{B}}_{2} \geq \frac{\epsilon^2}{4q\ell^2} - \nu(\lambda) \ .$$

□