Practical Leakage-Resilient Identity-Based Encryption from Simple Assumptions

Sherman Chow
New York University
schow@cs.nyu.edu

Yevgeniy Dodis
New York University
dodis@cs.nyu.edu

Yannis Rouselakis
The University of Texas at Austin
jrous@cs.utexas.edu

Brent Waters
The University of Texas at Austin
bwaters@cs.utexas.edu

ABSTRACT

We provide new constructions of *Leakage-Resilient Identity-Based Encryption* systems (IBE) in the Standard model. We apply a hash proof technique in the existing IBE schemes of Boneh-Boyen, Waters, and Lewko-Waters. As a result, we achieve leakage-resilience under the respective static assumptions of the original systems in the Standard model. The first two systems are secure under the simple *Decisional Bilinear Diffie-Hellman* assumption (DBDH).

The first system is selectively secure and serves as a stepping stone to construct the fully secure system. This second system is the first leakage-resilient fully secure system under DBDH in the Standard model. Finally the third system achieves full security with shorter public parameters but is based on three non-standard static assumptions.

The efficiency of our transformed systems is almost the same as the original ones.

1. INTRODUCTION

Traditionally in cryptography, we assume that the secret keys are completely hidden from the possible attackers. However several works [18, 19, 15] showed that this premise is not necessarily true in real systems. Many attacks such as timing attacks, power dissipation, cold-boot attacks, etc. can extract some bits of information from the secret keys or the state of the encrypting system, compromising security. In response to this, there has been a surge of interest in creating leakage-resilient cryptographic schemes [22, 2, 17]. Ideally, we would prefer systems resilient to large amounts of leakage with comparable efficiency to original systems and security based on simple assumptions in the Standard model.

One of the settings that cryptographers tried to implement leakage-resilience is the Identity-Based Encryption schemes (IBE) [24, 5, 7, 4, 25]. An IBE system gives the ability to different parties to encrypt messages knowing only the identity of the receiver. The identities are used in a way similar to public encryption keys and therefore we avoid the problem of public-key distribution. Recently Alwen et al. [1] gave exciting new leakage-resilient constructions for IBE from Lattice, Quadratic Residuosity (QR), and truncated augmented bilinear Diffie-Hellman exponent (q-TABDHE) assumptions using hash proof techniques. However the first two systems are secure in the Random Oracle model and the third, which is based on Gentry's IBE system [13], is secure under a complex "q-type" assumption, where the size of the assumption grows linearly with the number of attacker's queries. Given that there exist IBE systems secure under static simple assumptions in the Standard model [4, 25, 20], it is a natural question to ask whether we can give leakage-resilient versions of these systems.

Our Results Our first system is based on the Boneh-Boyen IBE [4]. This is only selectively-secure but it serves as a simpler version of the fully secure second system based on Waters IBE [25]. We prove that both systems are secure under the decisional Bilinear Diffie-Hellman assumption. This is a well-studied static assumption used many times in various constructions. However, the second system has large public parameters. In order to overcome this obstacle we present a third system based on the Lewko-Waters IBE [20] under three static assumptions related to composite order bilinear groups. These assumptions can be shown to hold in the generic group model if factoring is hard [20]. Efficiency results of the new systems compared to the old ones are shown in table 1.

The original systems used random secret keys with only one degree of freedom, which was explorable to the secret-key holder. This means that the owner of the secret key could re-randomize his key arbitrarily without knowing the secret parameters of the IBE system (the master secret key). In this sense the information each key holds is deterministic. The new technique we applied was to add another randomness to the secret keys, called "tag", coupled with some master secret key terms. As a result, the secret-key holder can not anymore re-randomize his key (in this degree of freedom). The added randomness allows the simulators of our security proofs to provide the attacker leaked information from a properly distributed secret key with a tag of our choice.

Obviously the ability of the simulator to create these secret keys allows him to decrypt the challenge ciphertext. One would ask then why is the attacker's response useful to the simulator. The answer is that we use a standard primitive in leakage-resilient constructions [22, 2] to "mask" the relationship between the leakage of the secret key and the ciphertext. This primitive, called *extractor* [23], makes it hard for any attacker to break the system given only a bounded amount of leakage from the secret key when the simulator "injects" the tag of the challenge identity to specific parts of

IBE System	Encryption	Decryption	Ciphertext Size	Public Parameters Size	Assumptions	Security
Boneh-Boyen [4]	$4 \cdot E$	$2 \cdot P$	$1 \cdot R_T + 2 \cdot R$	$1 \cdot R_T + 3 \cdot R$	DBDH	Selective
L-R BB, Section 3	$5 \cdot E$	$2 \cdot P$	$1 \cdot R_T + 2 \cdot R + 1 \cdot Ext$	$2 \cdot R_T + 3 \cdot R$	DBDH	Selective
Waters [25]	$3 \cdot E$	$2 \cdot P$	$1 \cdot R_T + 2 \cdot R$	$1 \cdot R_T + (B+2) \cdot R$	DBDH	Full
L-R W, Section 4	$4 \cdot E$	$2 \cdot P$	$1 \cdot R_T + 2 \cdot R + 1 \cdot Ext$	$2 \cdot R_T + (B+2) \cdot R$	DBDH	Full
Lewko-Waters [20]	$4 \cdot E$	$2 \cdot P$	$1 \cdot R_T + 2 \cdot R$	$1 \cdot R_T + 3 \cdot R$	1,2,3	Full
L-R LW, Section 5	$5 \cdot E$	$2 \cdot P$	$1 \cdot R_T + 2 \cdot R + 1 \cdot Ext$	$2 \cdot R_T + 3 \cdot R$	1,2,3	Full

Table 1: Efficiency results for existing systems and our constructions.
L-R denotes the leakage-resilient version of each system as presented in this paper. For encryption and decryption times we count only the dominant operations, which are the exponentiations in $\mathbb{G}$ and $\mathbb{G}_T$ denoted as E and the pairings denoted as P. For sizes we denote by R_T, R the number of bits for the representation of elements of $\mathbb{G}_T$ and $\mathbb{G}$, respectively. Ext is the size of plaintext messages plus the size of the extractor's seed. Typically the messages are symmetric encryption keys. B is the number of bits of each identity in the Waters' systems. Assumptions 1,2,3 are the assumptions in Section 2.6.2. Note that the times for each operation in the third system are larger than the respective ones in the first two because we use composite order bilinear groups instead of prime order groups.

the ciphertext.

1.1 Related Work

Identity-Based Encryption was first proposed in [24] and the first construction, secure in the Random Oracle model, was given in [5]. By utilizing a weaker notion of security, known as Selective security, many IBE systems were built in the Standard model [4, 7]. The first fully secure IBE system based on a simple assumption was given in [25].

Leakage-resilient systems on the other hand present more diversity and different models have been proposed. Micali and Reyzin [21] proposed a leakage model called "only computation leaks information", where unbounded amount of leakage is allowed but only from parts of memory that are accessed. A more general model of leakage, which captures the cold-boot memory attacks of [15], is the ability of the attacker to call an arbitrary leakage function on the secret key or the state of the encryption algorithm. Obviously the amount of leakage has to be bounded in this case, since an attacker can get the entire secret key. Two different models have been proposed: the Relative leakage model [22, 17, 9], where the leakage is a portion of the secret key and depends on the security parameter, and the Bounded Retrieval model [11, 8, 12, 1, 2], which allows for arbitrary large leakage and increasing sizes of secret keys, but with constant cost of encryption and decryption unrelated to the amount of leakage.

1.2 Organization of the paper

In Section 2 we present the leakage model we will use, the security definitions and the complexity assumptions. In Sections 3,4,5 we give the selectively secure, fully secure, and fully secure with short public parameters IBE systems, respectively. These sections start with the algorithms of each system and conclude with the proofs of security. In Section 4 we give only the construction. Full version of this paper can be found in `http://www.cs.utexas.edu/~jrous/`. Finally in Section 6 we provide some interesting open problems.

2. PRELIMINARIES

2.1 Notation

If $n \in \mathbb{N}$ then 1^n is a string of ones, i.e. n encoded in unary. $s \xleftarrow{R} S$ denotes that s is picked uniformly at random from the set S. For $n \in \mathbb{N}$ we write $[n]$ as shorthand for $\{1, 2, \cdots, n\}$ and $[n]_0$ for $[n] \cup \{0\}$. We write PPT for probabilistic polynomial time. By $\mathsf{negl}(n)$ we denote a negligible function of n, i.e. a function $f : \mathbb{N} \to \mathbb{R}$ such that for every $c > 0$ and an infinite number of n's: $f(n) \leq n^{-c}$.

2.2 Identity-Based Encryption

An Identity-Based Encryption scheme [24] consists of four PPT algorithms:

Setup$(1^n) \to (PP, MSK)$ The setup algorithm takes a security parameter n as input, and outputs a set of public parameters PP and a master secret key MSK. The security parameter is encoded in unary, so that all algorithms run in polynomial time in n. The remaining algorithms take implicitly the security parameter 1^n and the public parameters PP as inputs.

KeyGen$(MSK, I) \to SK$ The key generation algorithm takes the master secret key, and an identity I as inputs, and outputs a private key SK for identity I.

Encrypt$(I, M) \to CT$ The encryption algorithm takes a message M to be encrypted, and an identity I as inputs. It outputs an encryption CT of the message M for identity I.

Decrypt$(SK, CT) \to M$ The decryption algorithm takes a secret key of identity I, and a ciphertext CT. It outputs the message M if the ciphertext was a correct encryption for identity I.

2.3 Leakage Model

In all constructions we use the Relative Leakage model on the secret keys, similar to [22], which allows the attacker to call an arbitrary function on the secret key of the identity he wants to attack with the restriction that the total output size is less than a relative bound. Our model is modified suitably for the IBE setting. As a result the attacker can get a leakage of up to $l = l(n)$ bits from one key of each identity. All leakage-resilient IBE systems have this restriction [1, 22, 2]: that from each identity only one secret key is produced. We allow the production of multiple secret keys but we require that the leakage comes from only one.

We denote by $\mathcal{I}$ the identity space and by $\mathcal{SK}$ the secret keys' space. The leakage is accessible to the attacker via calls to a $\mathsf{Leak}(I, h_i)$ method, where $I \in \mathcal{I}$ and $h_i : \mathcal{SK} \to \{0,1\}^{l_i}$. The challenger has to check that the sum of all l_i's for the same identity does not exceed the leakage bound l and if this is true, he has to return the value of the function $h_i(SK_I)$ on a secret key of identity I. This secret key has to be the same in all subsequent calls to $\mathsf{Leak}(I, \cdot)$. If the requested leakage exceeds l then the challenger ignores the query. Since we care only about PPT attackers, we require that the total number of queries is polynomial in n and that all h_i's are efficiently computable.

2.4 Security Definition

To define leakage security, we modify the usual IbeCpa security game [14, 5] appropriately. The important change is that the adversary-attacker can get leakage from many secret keys of identities that eventually he will not forge on. The challenger has to keep track of all these leakage queries, so that if the adversary gets access to the entire secret key for an identity, he can not choose this as the challenge identity.

In the IbeCpa security game the adversary has the ability to make KeyGen queries to the challenger for any identity other than the challenge identity. In the new security game the adversary can make two additional queries, called Leak and Reveal. The Reveal query encodes the will of the attacker to learn the entire secret key of a previous leakage query. Obviously the identity of this key can not be used later as the challenge identity. We refer to both of these queries as *leakage queries*.

All the above queries can happen adaptively, i.e. they can depend on previous ones. However for all adversaries w.l.o.g. we will make some assumptions on the way they work:

We assume that after a Reveal or a KeyGen query on identity I, no Leak queries on I are made. This does not restrict the adversary since after such queries all secret keys on this identity are essentially "unlocked" and he can have access to all of their bits. Therefore he can compute the intended leakage himself.

We assume that just before the Challenge phase all secret keys leaked, except the one for the challenge identity I^*, are revealed to the adversary. Obviously this gives more power to him.

Finally, we assume that the adversary has made at least one leakage query for the challenge identity. Again this gives more power to him. With this assumption we know that the challenge identity is in the list of leaked keys; a fact that will be used in the security proofs.

The new security game, called CpaLeak, is parameterized by a security parameter n and a leakage parameter $l = l(n)$ and consists of the following phases:

Setup The challenger runs the $\mathsf{Setup}(1^n)$ algorithm and obtains the public parameters PP and the master secret key MSK. It gives PP to the adversary.

Also he creates two sets $\mathcal{K} = \emptyset$ and $\mathcal{L} = \emptyset$ to keep track of the keygen and the leakage queries, respectively. Both $\mathcal{K}$ and $\mathcal{L}$ are sets of triples of identities, secret keys, and a counter, i.e. $(I, SK, Cntr) \in \mathcal{I} \times \mathcal{SK} \times \mathbb{N}$.

Phase 1 The adversary can make one of the following three queries to the challenger:

1. A query $\mathsf{Leak}(I, h_i)$ where $h_i : \mathcal{SK} \to \{0,1\}^{l_i}$. The challenger checks if there is a triple $(I, SK, Cntr)$ in $\mathcal{L}$. If it is not, he runs the $\mathsf{KeyGen}(MSK, I) \to SK$ algorithm and inserts $(I, SK, 0)$ to $\mathcal{L}$. After this step or if the triple was in $\mathcal{L}$, he checks if $Cntr + l_i \leq l$. If this it true, he responds with $h_i(SK)$ and sets $Cntr \leftarrow Cntr + l_i$ in $(I, SK, Cntr)$. Otherwise he responds with a dummy value $\perp$.

2. A query $\mathsf{Reveal}(I)$ where I is an identity. The challenger checks if there is a triple $(I, SK, Cntr)$ in $\mathcal{L}$. If it is then he moves it to $\mathcal{K}$ and gives SK to the adversary. If it is not in $\mathcal{L}$ but in $\mathcal{K}$ (this means that another reveal query has been issued before), then he reveals SK, as well. Finally if it is not in any of these sets, the challenger creates a new secret key by invoking $\mathsf{KeyGen}(MSK, I) \to SK$, puts the triple $(I, SK, 0)$ to set $\mathcal{K}$, and responds with SK as above.

3. A query $\mathsf{KeyGen}(MSK, I)$ where the challenger returns a fresh secret key for identity I and moves, if it exists, the triple $(I, SK, Cntr)$ from set $\mathcal{L}$ to $\mathcal{K}$.

Challenge The adversary submits two messages M_0, M_1 of equal size and a challenge identity I^*, with the restriction that $(I^*, \cdot, \cdot) \notin \mathcal{K}$. The challenger sets a bit $\sigma \overset{R}{\leftarrow} \{0,1\}$ uniformly at random and encrypts M_σ under I^*. He sends the resulting ciphertext CT^* to the adversary.

Phase 2 This is the same as Phase 1 with the restriction that no leakage queries are allowed; only KeyGen on identities different than I^*. [1]

Guess The adversary outputs a bit σ'. We say that he succeeds if $\sigma' = \sigma$.

The advantage of an adversary $\mathcal{A}$ on breaking an IBE scheme Π with security parameter n and leakage l is defined as $\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},\Pi}(n, l) = \Pr[\mathcal{A} \text{ succeeds}] - \frac{1}{2}$.

DEFINITION 2.1. *An IBE scheme Π is l-leakage fully secure if for all PPT adversaries $\mathcal{A}$*

$$\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},\Pi}(n, l) \leq \mathsf{negl}(n)$$

If we modify the above security game so that the adversary gives the challenge identity I^* to the challenger before the setup phase, we get the game SelLeak in order to define selective security. Notice that in this case the challenger doesn't have to keep track of the leaked and keygen-ed keys since he knows from the beginning what is the challenge identity; the set $\mathcal{L}$ contains only $(I^*, \cdot, \cdot)$.

If $\mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi}(n) = \Pr[\mathcal{A} \text{ succeeds}] - \frac{1}{2}$ as before, we have:

DEFINITION 2.2. *An IBE scheme Π is l-leakage selectively secure if for all PPT adversaries $\mathcal{A}$*

$$\mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi}(n, l) \leq \mathsf{negl}(n)$$

In all lemmas and definitions we will not write the dependence of Adv in n, l for easiness of notation.

2.5 Min-Entropy - Extractors

In our constructions we will use some of the following notions and primitives. For a detailed treatment see [10, 23].

The min-entropy of a random variable X is defined as $\mathbf{H}_\infty(X) = -\log(\max_x \Pr[X = x])$. We will mainly use the

[1] The reason we don't allow Leak queries in Phase 2 is that after the challenge phase the adversary can encrypt the entire decryption algorithm of CT^* as a function on the secret key of I^*. Then obviously he can always win the game. Also there is no need for Reveal queries since before the challenge phase all leaked keys are revealed.

average min-entropy of a random variable X conditioned on another random variable Y. This is defined as

$$\tilde{\mathbf{H}}_\infty(X|Y) = -\log\left(\mathbb{E}_{y\leftarrow Y}\left[\max_x \Pr[X=x|Y=y]\right]\right)$$

where $\mathbb{E}_{y\leftarrow Y}$ denotes the expected value over all values of Y.

Intuitively, min-entropy of a random variable measures the difficulty of any adversary (even an unbounded one) to predict the value of the variable. Higher values of min-entropy imply lower success probability for the adversary, because the best strategy is to predict the value with the highest probability: $\max_x \Pr[X=x]$. The average min-entropy captures the same notion of predicting a variable X given knowledge of another random variable Y.

The following lemma was proved in [10] regarding average min-entropy:

LEMMA 2.3. *For any random variables A, B, C such that B has 2^l possible values, we have that $\tilde{\mathbf{H}}_\infty(A|(B,C)) \geq \tilde{\mathbf{H}}_\infty(A|C) - l$.*

The statistical distance between two random variables X, Y over a finite domain Ω is defined as

$$\mathbf{SD}(X,Y) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X=\omega] - \Pr[Y=\omega]|$$

The use of the following functions, called extractors, is typical in leakage-resilient systems [2, 1, 22]:

DEFINITION 2.4. *A polynomial-time function $\mathsf{ext} : \mathbb{G} \times \{0,1\}^\mu \to \{0,1\}^m$ is an average-case (h,ϵ)-strong extractor if for all pairs of random variables (X,Y) such that $X \in \mathbb{G}$ and $\tilde{\mathbf{H}}_\infty(X|Y) \geq h$, we have that*

$$\mathbf{SD}((\mathsf{ext}(X,U_\mu),U_\mu,Y),(U_m,U_\mu,Y)) \leq \epsilon$$

where $\mathbb{G}$ is a non-empty set, and U_μ, U_m are two uniformly distributed random variables over $\{0,1\}^\mu$, $\{0,1\}^m$ respectively.

In [10] Dodis et al. proved the following lemma which gives an explicit construction of an average-case strong extractor:

LEMMA 2.5. *Let A, B be random variables such that $A \in \mathbb{G}$ and $\tilde{\mathbf{H}}_\infty(A|B) \geq h$. Let $\mathcal{H} = \{H_s : \mathbb{G} \to \{0,1\}^m\}_{s\in S}$ be a family of universal hash functions. If $m \leq h - 2\log\left(\frac{1}{\epsilon}\right) + 2$ then*

$$\mathbf{SD}\left((H_{U_S}(A), U_S, B), (U_m, U_S, B)\right) \leq \epsilon$$

where U_S is a uniformly distributed variable over S.

2.6 Assumptions

We present four assumptions in this section. We use the first one to prove security of the first two systems in Sections 3,4 and the other three for the last system in Section 5.

2.6.1 Decisional Bilinear Diffie-Hellman

The security of the first two IBE systems will be based on the well-known Decisional Bilinear Diffie-Hellman assumption (DBDH). This is defined via the following game:

Given a group generator algorithm $\mathcal{G}(1^n)$ the challenger generates a group $\mathbb{G}$ of prime order $p = \Theta(2^n)$ which admits an efficiently computable non-degenerate bilinear map[2] $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The challenger picks uniformly at random a generator $g \xleftarrow{R} \mathbb{G}$ and four random exponents $x, y, z, w \xleftarrow{R} \mathbb{Z}_p$. He computes g^x, g^y, g^z, $T_0 = e(g,g)^{xyz}$, and $T_1 = e(g,g)^{xyw}$. We denote by D the tuple

$$D = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, g^y, g^z)$$

He then flips a random coin $\nu \xleftarrow{R} \{0,1\}$ and gives to the adversary the tuple (D, T_ν). That is, either the element $e(g,g)^{xyz}$ or a random element in $\mathbb{G}_T$.

The adversary outputs a guess $\nu' \in \{0,1\}$. We say that he succeeds if $\nu' = \nu$. The advantage of an adversary $\mathcal{A}$ on $\mathcal{G}(1^n)$ is defined as:

$$\mathsf{Adv}^{\mathsf{DBDH}}_{\mathcal{G},\mathcal{A}}(n) = \Pr[\mathcal{A} \text{ succeeds}] - \frac{1}{2} =$$

$$= \frac{1}{2}\left(\Pr[\mathcal{A}(D, e(g,g)^{xyz}) = 0] - \Pr[\mathcal{A}(D, e(g,g)^{xyw}) = 0]\right)$$

where $D = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, g^y, g^z)$.

ASSUMPTION 2.6 (DBDH). *We say that $\mathcal{G}$ satisfies the DBDH assumption if for all PPT adversaries $\mathcal{A}$*

$$\mathsf{Adv}^{\mathsf{DBDH}}_{\mathcal{G},\mathcal{A}}(n) \leq \mathsf{negl}(n)$$

2.6.2 Composite Order Bilinear Groups

In the last IBE system we will use composite order bilinear groups introduced in [6]. As before the group generator algorithm $\mathcal{G}$ takes as input a security parameter 1^n and outputs the description of a bilinear group $\mathbb{G}$ of order $N = p_1p_2p_3$ where p_1, p_2, p_3 are three prime numbers of magnitude $\Theta(2^n)$. We assume that the generator algorithm outputs the values of p_1, p_2, p_3 and generators of the respective subgroups.

We let $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ denote these subgroups. We also let $\mathbb{G}_{p_ip_j}$ denote the subgroup of order p_ip_j. We can prove that when $h_i \in \mathbb{G}_{p_i}$ and $h_j \in \mathbb{G}_{p_j}$ for $i \neq j$, we have that $e(h_i, h_j) = 1_{\mathbb{G}_T}$ [20].

We define the following games:

Game 1 In this game the challenger runs $\mathcal{G}(1^n)$ and gives to the adversary $\mathcal{A}$ the tuple $D^1 = (N, \mathbb{G}, \mathbb{G}_T, e, g_{p_1}, g_{p_3})$.

Then the challenger flips a random coin $\nu \xleftarrow{R} \{0,1\}$ and picks $(a,b) \xleftarrow{R} \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2}$. He computes $T_0^1 = g_{p_1}^a$ and $T_1^1 = g_{p_1}^a g_{p_2}^b$ and sends T_ν^1 to $\mathcal{A}$.

In the end $\mathcal{A}$ outputs a bit ν', and succeeds if $\nu' = \nu$.

Game 2 In this game the challenger runs $\mathcal{G}(1^n)$ and picks random exponents $(x_1, x_2, x_3) \xleftarrow{R} \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \mathbb{Z}_{p_3}$. He gives to $\mathcal{A}$ the tuple

$$D^2 = (N, \mathbb{G}, \mathbb{G}_T, e, g_{p_1}, g_{p_3}, g_{p_1}^{x_1} g_{p_2}, g_{p_2}^{x_2} g_{p_3}^{x_3})$$

Then he flips a random coin $\nu \xleftarrow{R} \{0,1\}$ and picks $(a,b,c) \xleftarrow{R} \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \mathbb{Z}_{p_3}$. He computes $T_0^2 = g_{p_1}^a g_{p_3}^c$ and $T_1^2 = g_{p_1}^a g_{p_2}^b g_{p_3}^c$ and sends T_ν^2 to $\mathcal{A}$.

$\mathcal{A}$ outputs a bit ν', and succeeds if $\nu' = \nu$.

Game 3 In this game the challenger runs $\mathcal{G}(1^n)$ and picks random exponents $(x_2, y_2, \beta, z) \xleftarrow{R} \mathbb{Z}_{p_2} \times \mathbb{Z}_{p_2} \times \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_1}$. He gives to $\mathcal{A}$ the tuple

$$D^3 = (N, \mathbb{G}, \mathbb{G}_T, e, g_{p_1}, g_{p_3}, g_{p_1}^{\beta} g_{p_2}, g_{p_1}^{z} g_{p_2}^{x_2}, g_{p_2}^{y_2})$$

Then he flips a random coin $\nu \xleftarrow{R} \{0,1\}$ and picks $w \xleftarrow{R} \mathbb{Z}_{p_1}$. He computes $T_0^3 = e(g_{p_1}, g_{p_1})^{\beta z}$ and $T_1^3 = e(g_{p_1}, g_{p_1})^{\beta w}$ and sends T_ν^3 to $\mathcal{A}$.

[2] We require that for every generator $g \in \mathbb{G}$ we have that $e(g,g) \neq 1$ and for every $a, b \in \mathbb{Z}_p : e(g^a, g^b) = e(g,g)^{ab}$.

$\mathcal{A}$ outputs a bit ν', and succeeds if $\nu' = \nu$.

Assumptions The advantage of any PPT adversary $\mathcal{A}$ in game i where $i \in \{1, 2, 3\}$ is:

$$\mathsf{Adv}^i_{\mathcal{G},\mathcal{A}}(n) = \frac{1}{2}\left(\Pr[\mathcal{A}(D^i, T^i_0) = 0] - \Pr[\mathcal{A}(D^i, T^i_1) = 0]\right)$$

We say that $\mathcal{G}$ satisfies *Assumption* i if for all PPT algorithms $\mathcal{A}$

$$\mathsf{Adv}^i_{\mathcal{G},\mathcal{A}}(n) \leq \mathsf{negl}(n)$$

3. OUR FIRST SYSTEM

Our first system is similar to the Boneh-Boyen IBE [4] and its security is based on the same assumption (DBDH).

We modified the Boneh-Boyen system by adding two secret parameters $x, y \in \mathbb{Z}_p$ (it already has $a, b \in \mathbb{Z}_p$) and an integer tag t in every secret key. The terms x, y will be used as the exponents from DBDH assumption and the resulting new term $e(g,g)^{xyz}$ in the ciphertext will serve as the challenge term of DBDH. The tag is used as a "trapdoor" for the simulator of our proofs to relate a, b to x, y in a way hidden from any attacker without knowing x, y. He picks a known tag t^* and sets $a = xt^* + \tilde{a}$ and $b = y + \tilde{b}$. Knowledge of t^* allows him to create secret keys of the challenge identity I^* tagged only with t^*. However, this key will seem random to the attacker; as if it was sampled from the entire space of $\mathcal{SK}_{I^*}$. Leakage-resilience comes in the end from the application of the extractor to $e(g,g)^{abz}$ which is transformed to a random term in case a non-valid DBDH term $e(g,g)^{xyw}$ is given. The same intuition holds for all three systems in this paper.

3.1 Construction

We will denote this system by Π. It consists of the following algorithms:

Setup The setup algorithm uses a group generator $\mathcal{G}(1^n)$ to create a bilinear group $\mathbb{G}$ of prime order p, as the one used in the DBDH assumption. It then picks uniformly at random $a, b, x, y \xleftarrow{R} \mathbb{Z}_p$ and $g, u, h \xleftarrow{R} \mathbb{G}$.

Let $l = l(n)$ be an upper bound on the amount of leakage. Then it sets an average-case $(\log|\mathbb{G}_T| - l, \epsilon_{\mathsf{ext}})$-strong extractor function $\mathsf{ext} : \mathbb{G}_T \times \{0,1\}^\mu \to \{0,1\}^m$. We assume that the message space is $\mathcal{M} = \{0,1\}^m$.

Finally it publishes the public parameters

$$PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, u, h, e(g,g)^{xy}, e(g,g)^{ab}, \mathsf{ext})$$

and stores privately the master secret key $MSK = (g^{xy}, g^{ab})$.

KeyGen(MSK, I) The key generation algorithm picks uniformly at random two exponents $t, r \xleftarrow{R} \mathbb{Z}_p$. Exponent t is meant to serve as a tag on the generated key. We assume that the identity space is $\mathcal{I} = \mathbb{Z}_p$. Then the algorithm calculates the following secret key for identity I:

$$SK = (s_1, s_2, s_3) = (t, g^{ab} g^{-xyt} (u^I h)^r, g^{-r})$$

Encrypt(I, M) The encryption algorithm picks uniformly at random an exponent $z \xleftarrow{R} \mathbb{Z}_p$ and a random seed $s \xleftarrow{R} \{0,1\}^\mu$ for the extractor function. The ciphertext is:

$$\begin{aligned} CT &= (c_1, c_2, c_3, c_4, c_5) \\ &= \left(M \oplus \mathsf{ext}\left(e(g,g)^{abz}, s\right), s, (u^I h)^z, g^z, e(g,g)^{xyz}\right) \end{aligned}$$

where by $\oplus$ we denote the bit-wise XOR operation.

Decrypt The decryption algorithm returns:

$$\mathsf{ext}\left(e(s_2, c_4) e(s_3, c_3) c_5^{s_1}, c_2\right) \oplus c_1$$

If the identities of the secret key and the ciphertext are equal, we get:

$$\begin{aligned} &\mathsf{ext}\left(e(s_2, c_4) e(s_3, c_3) c_5^{s_1}, c_2\right) \oplus c_1 = \\ &= \mathsf{ext}(e(g,g)^{abz} e(g,g)^{-xyzt} \cdot \\ &\quad \cdot e(u^I h, g)^{rz} e(g, u^I h)^{-rz} e(g,g)^{xyzt}, s) \oplus \\ &\quad \oplus M \oplus \mathsf{ext}\left(e(g,g)^{abz}, s\right) = M \end{aligned}$$

which proves correctness of decryption.

3.2 Security

To prove selective security of our IBE system we define a new security game which is called SelModfd. This game is the same as the SelLeak security game except the challenge phase. In the challenge phase of this game the challenger does not execute the correct encryption on M_b but a new "encryption" scheme.

In this phase he picks uniformly at random two exponents $z, w \xleftarrow{R} \mathbb{Z}_p$ and a random bit $\sigma \xleftarrow{R} \{0,1\}$. He computes the following:

$$\begin{aligned} C &= e(g,g)^{abz} e(g,g)^{xy(w-z)t^*} & c^*_2 &\xleftarrow{R} \{0,1\}^\mu \\ c^*_1 &= M_\sigma \oplus \mathsf{ext}(C, c_2) & c^*_3 &= (u^{I^*} h)^z \\ c^*_4 &= g^z & c^*_5 &= e(g,g)^{xyw} \end{aligned}$$

where t^* is the tag of the secret key SK_{I^*} of the challenge identity leaked to the adversary during Phase 1. Remember our assumption that some information is always leaked from that secret key. The challenge ciphertext returned to the adversary is $(c^*_1, c^*_2, c^*_3, c^*_4, c^*_5)$.

With some algebraic manipulation it is easy to see that the above ciphertext always decrypts with a secret key of I^* only if its tag is t^*. Therefore the above is not a valid encryption algorithm. However we can define by $\mathsf{Adv}^{\mathsf{SelModfd}}_{\mathcal{A},\Pi}(n, l) = \Pr[\mathcal{A} \text{ succeeds}] - \frac{1}{2}$ the advantage of any algorithm $\mathcal{A}$ in this game.

Intuition of the proof Using the original a, b parameters the Boneh-Boyen system naturally allows the cancellation of the ab exponent in the creation of secret keys for identities different than I^*. Notice that if the simulator knows g^{ab}, he can solve the DBDH assumption himself [4]. Thus this cancellation is necessary in the reduction. However in our setting the simulator has to create one secret key for I^*, which is not possible in the original system. In order to do this we added two more parameters x, y and the tag t. Now the DBDH parameters are x and y. As we said terms a and b depend (obliviously to the attacker) on x, y and t^*. Using t^* the simulator can create a secret key of I^* since the resulting g^{xyt^*}, g^{-xyt^*} terms cancel out. The original trick of Boneh-Boyen still works for identities other than I^* for x, y.

Finally, in the SelModfd game the blinding factor has enough entropy due to the extractor, because without knowledge of leakage the attacker sees a random term. Getting leakage of l bits allows him to decrease the entropy at most by l and the extractor's parameters ensure that the blinding factor is close to uniform.

LEMMA 3.1. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi} - \mathsf{Adv}^{\mathsf{SelModfd}}_{\mathcal{A},\Pi} = \epsilon$$

Then we can build a PPT algorithm $\mathcal{B}$ with advantage $\epsilon/2$ in breaking the DBDH assumption.

PROOF. $\mathcal{B}$ takes in a DBDH challenge

$$(p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, g^y, g^z, T_\nu)$$

According to SelLeak game, $\mathcal{A}$ gives to $\mathcal{B}$ an identity I^* that he wishes to forge on.

For the Setup phase, $\mathcal{B}$ sets implicitly $a = xt^* + \tilde{a}$ and $b = y + \tilde{b}$, where all new variables such as $t^*, \tilde{a}, \tilde{b}$ are chosen uniformly at random from $\mathbb{Z}_p$ from this point on. Also he sets $u = g^x g^{\tilde{u}}$ and $h = g^{-xI^*} g^{\tilde{h}} = (g^x)^{-I^*} g^{\tilde{h}}$. He computes $e(g,g)^{ab} = e(g^x, g^y)^{t^*} e(g^x, g)^{t^*\tilde{b}} e(g, g^y)^{\tilde{a}} e(g,g)^{\tilde{a}\tilde{b}}$ and $e(g,g)^{xy} = e(g^x, g^y)$. Also, $\mathcal{B}$ picks a suitable extractor function ext.

We argue that the variables a, b, u, h are properly distributed because $\mathcal{B}$ picks the random exponents $\tilde{a}, \tilde{b}, \tilde{u}, \tilde{h}$ uniformly at random. The elements g, x, y are supposed to be picked uniformly at random by the DBDH challenger from their respective groups.

Therefore, the view of $\mathcal{A}$ is completely legitimate and $\mathcal{B}$ responds with the public parameters

$$PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, u, h, e(g,g)^{xy}, e(g,g)^{ab}, \mathsf{ext})$$

For Phase 1, $\mathcal{B}$ has to calculate one secret key for I^* for the leakage queries and an arbitrary number of secret keys for all other identities.

For queries on $I \neq I^*$, he sets $s'_1 = t^* - \tilde{t}$, where $\tilde{t} \xleftarrow{R} \mathbb{Z}_p$. Therefore s'_1 is properly distributed and it is a known term. Also he sets implicitly $r = -y\tilde{t}/(I - I^*) + \tilde{r}$. Since $\tilde{r} \xleftarrow{R} \mathbb{Z}_p$, r is properly distributed.

Then $\mathcal{B}$ computes

$$\begin{aligned}
s'_2 &= g^{ab} g^{-xyt} (u^I h)^r \\
&= g^{xyt^*} g^{xt^*\tilde{b}} g^{y\tilde{a}} g^{\tilde{a}\tilde{b}} \cdot g^{-xyt^*} g^{xy\tilde{t}} \cdot \\
&\cdot \left(g^{x(I-I^*)} g^{\tilde{u}I} g^{\tilde{h}}\right)^{-y\tilde{t}/(I-I^*)} \cdot \left(g^{x(I-I^*)} g^{\tilde{u}I} g^{\tilde{h}}\right)^{\tilde{r}} \\
&= (g^x)^{t^*\tilde{b}} (g^y)^{\tilde{a}} g^{\tilde{a}\tilde{b}} \cdot \\
&\cdot (g^y)^{-\tilde{u}I\tilde{t}/(I-I^*)} (g^y)^{-\tilde{h}\tilde{t}/(I-I^*)} (g^x)^{\tilde{r}(I-I^*)} g^{\tilde{r}\tilde{u}I} g^{\tilde{r}\tilde{h}} \\
s'_3 &= g^{-r} = (g^y)^{\tilde{t}/(I-I^*)} g^{-\tilde{r}}
\end{aligned}$$

and responds to all leakage and keygen queries with (s'_1, s'_2, s'_3).

For a Leak query on I^*, he picks an exponent $r \xleftarrow{R} \mathbb{Z}_p$ and sets $s^*_1 = t^*$. Since t^* was chosen randomly in the Setup phase, s^*_1 is properly distributed.

Then he computes

$$\begin{aligned}
s^*_2 &= g^{ab} g^{-xyt^*} (u^{I^*} h)^r = g^{xyt^*} g^{xt^*\tilde{b}} g^{y\tilde{a}} g^{\tilde{a}\tilde{b}} \cdot \\
&\qquad \cdot g^{-xyt^*} \cdot g^{xI^* r} g^{\tilde{u}I^* r} g^{-xI^* r} g^{\tilde{h}r} \\
&= (g^x)^{t^*\tilde{b}} (g^y)^{\tilde{a}} g^{\tilde{a}\tilde{b}} g^{\tilde{u}I^* r} g^{\tilde{h}r} \\
s^*_3 &= g^{-r}
\end{aligned}$$

and answers all Leak queries using $SK_{I^*} = (s^*_1, s^*_2, s^*_3)$.

Notice that by choosing different $\tilde{t}$'s $\mathcal{B}$ can calculate many secret keys on I with different tags since $t = t^* - \tilde{t}$, while he can generate secret keys with only one tag for I^*.

In the Challenge phase, $\mathcal{A}$ submits two messages M_0, M_1 to $\mathcal{B}$. Then $\mathcal{B}$ flips a random coin $\sigma \xleftarrow{R} \{0,1\}$ and computes the following "encryption" of M_σ:

$$\begin{aligned}
&c^*_2 \xleftarrow{R} \{0,1\}^\mu && c^*_3 = (u^{I^*} h)^z = (g^z)^{\tilde{u}I^*} (g^z)^{\tilde{h}} \\
&c^*_4 = g^z && c^*_5 = T_\nu \\
&C = e(s^*_2, c^*_4) e(s^*_3, c^*_3) (c^*_5)^{s^*_1} && c^*_1 = M_\sigma \oplus \mathsf{ext}(C, c^*_2)
\end{aligned}$$

Remember that T_ν is the challenge term given by the DBDH challenger.

For Phase 2, $\mathcal{B}$ calculates the requested secret keys as he did in Phase 1. At the Guess phase, $\mathcal{A}$ outputs a guess σ'. Then $\mathcal{B}$ returns to the DBDH challenger $\nu' = 0$ if $\sigma' = \sigma$ or $\nu' = 1$ if $\sigma' \neq \sigma$.

We will prove that the advantage of $\mathcal{B}$ in breaking the DBDH assumption is $\epsilon/2$. To see this, notice that if $T_\nu = e(g,g)^{xyz}$ the challenge ciphertext is a correct ciphertext according to our original encryption scheme. That is because the term $e(s^*_2, c^*_4) e(s^*_3, c^*_3) (c^*_5)^{s^*_1} = e(g,g)^{abz}$ as one can easily verify. Thus the probability that $\mathcal{A}$ succeeds in the game is exactly $\frac{1}{2} + \mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi}$. Since $\mathcal{B}$ outputs 0 when $\mathcal{A}$ succeeds we get that

$$\Pr[\mathcal{B}(D, e(g,g)^{xyz}) = 0] = \frac{1}{2} + \mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi}$$

where $D = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, g^y, g^z)$.

On the other hand if $T_\nu = e(g,g)^{xyw} = c^*_5$ then $C = e(g,g)^{abz} e(g,g)^{xy(w-z)t^*}$ and $\mathcal{A}$ plays the modified game. Therefore we have that

$$\Pr[\mathcal{B}(D, e(g,g)^{xyw}) = 0] = \frac{1}{2} + \mathsf{Adv}^{\mathsf{SelModfd}}_{\mathcal{A},\Pi}$$

Combining the above equations and using lemma's assumption, we get that the advantage of $\mathcal{B}$ in DBDH is:

$$\begin{aligned}
&\frac{1}{2}\left(\Pr[\mathcal{B}(D, e(g,g)^{xyz}) = 0] - \Pr[\mathcal{B}(D, e(g,g)^{xyw}) = 0]\right) = \\
&= \frac{1}{2}\left(\mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi} - \mathsf{Adv}^{\mathsf{SelModfd}}_{\mathcal{A},\Pi}\right) = \frac{\epsilon}{2}
\end{aligned}$$

□

LEMMA 3.2. *For any PPT adversary $\mathcal{A}$ it holds that*

$$\mathsf{Adv}^{\mathsf{SelModfd}}_{\mathcal{A},\Pi} \leq 2\epsilon_{\mathsf{ext}}$$

PROOF. In the modified game, it is true that

$$C = e(g,g)^{abz} e(g,g)^{xy(w-z)\cdot t^*}$$

where t^* is the tag of the secret key for I^*. If we assume that this secret key is hidden from the adversary, then C is distributed uniformly at random in $\mathbb{G}_T$. That is, because $w \neq z \bmod p$ and t^* lies only in this specific secret key. [3]

Suppose we define by Z the set of all terms (public parameters, secret keys, challenge ciphertext) given to the adversary $\mathcal{A}$ except the leakage, the random seed c^*_2, and the c^*_1 part of the challenge ciphertext. Then according to the above argument $\tilde{\mathbf{H}}_\infty(C|Z) = \log|\mathbb{G}_T|$. But the attacker has access to at most l bits of leakage from the secret key, i.e. to a random variable Y with 2^l values. By lemma 2.3 we know that

$$\tilde{\mathbf{H}}_\infty(C|(Y,Z)) \geq \tilde{\mathbf{H}}_\infty(C|Z) - l = \log|\mathbb{G}_T| - l$$

[3] $w = z$ only with negligible probability in n.

Therefore the definition of a $(\log|\mathbb{G}_T| - l, \epsilon_{\mathsf{ext}})$ - strong extractor asserts that

$$\mathbf{SD}\left((\mathsf{ext}(C,S),S,Y,Z),(U_m,S,Y,Z)\right) \leq \epsilon_{\mathsf{ext}}$$

where S is the random variable for the seed $c_2^* \in \{0,1\}^\mu$ distributed uniformly at random. Notice that S, Y, Z are the values of all the random variables known to the adversary: seed, leakage, and the rest, respectively.

Thus the statistical distance of $c_1^* = M_\sigma \oplus \mathsf{ext}\,(C, c_2^*)$ from the uniform distribution is at most ϵ_{ext} for each σ. Therefore the statistical distance between the two possible ciphertexts is at most $2\epsilon_{\mathsf{ext}}$ and no adversary (even an unbounded one) can distinguish them with advantage more than this. □

THEOREM 3.3. *If the DBDH assumption holds and the extractor's second parameter ϵ_{ext} used in Π is negligible in n, then system Π is l-leakage selectively secure, where $l = \log|\mathbb{G}_T| - k$ and k is the extractor's first parameter.*

PROOF. Suppose ϵ_{DBDH} is the maximum advantage of all PPT adversaries in the DBDH game. Then according to lemmas 3.1 and 3.2, for any PPT adversary $\mathcal{A}$ we have that:

$$\mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi} - \mathsf{Adv}^{\mathsf{SelModfd}}_{\mathcal{A},\Pi} \leq 2\epsilon_{DBDH} \implies$$
$$\mathsf{Adv}^{\mathsf{SelLeak}}_{\mathcal{A},\Pi} \leq 2\left(\epsilon_{DBDH}(n) + \epsilon_{\mathsf{ext}}(n)\right)$$

If the premises of the theorem hold both $\epsilon_{DBDH}(n)$ and $\epsilon_{\mathsf{ext}}(n)$ are negligible functions of n. □

Leakage Summary In order to give a concrete bound on the leakage allowed by our scheme we use lemma 2.5 which states that efficient constructions of extractors exist whenever $m \leq \log|\mathbb{G}_T| - l - 2\log(1/\epsilon_{\mathsf{ext}}) + 2$. Since we require $\epsilon_{\mathsf{ext}}(n)$ to be negligible, we allow leakage of up to $l = \log p - \omega(\log n) - m$ bits. If we assume that we can represent elements of $\mathbb{Z}_p$ and $\mathbb{G}$ with $\log p$ bits, the fraction of the secret key we allow to be leaked is slightly less than $1/3$.

4. OUR SECOND SYSTEM

In this section we will modify the existing system in a way similar to [25], to get a fully secure leakage-resilient IBE scheme. The only change is that we use the Waters' hash $u_0\prod_{i=1}^B u_i^{I_i}$ for identities $I \in \{0,1\}^B$ instead of the Boneh-Boyen hash $u^I h$ for $I \in \mathbb{Z}_p$.

4.1 Construction

We will denote the new system by P. The algorithms are the following:

Setup This works the same way as the **Setup** algorithm of Π with the difference that instead of picking random $u, h \xleftarrow{R} \mathbb{G}$, it picks $B+1$ uniform random elements of $\mathbb{G}$, $u_0, u_1, \cdots, u_B \xleftarrow{R} \mathbb{G}$, where $B = B(n)$ is a polynomial in the security parameter n. We assume that all identities are B-bit vectors, and we denote the i-th bit of I by I_i.

The public parameters are

$$PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, u_0, u_1, \cdots, u_B, e(g,g)^{xy}, e(g,g)^{ab}, \mathsf{ext})$$

and the master secret key $MSK = (g^{xy}, g^{ab})$.

Algorithms **KeyGen** and **Encrypt** are the same as the respective ones from Π if we substitute the $u^I h$ term with the Waters' hash $u_0\prod_{i=1}^B u_i^{I_i}$ for $I \in \{0,1\}^B$. **Decrypt** is exactly the same as Π:

KeyGen(MSK, I)

$$SK = (s_1, s_2, s_3) = (t, g^{ab}g^{-xyt}(u_0\prod_{i=1}^B u_i^{I_i})^r, g^{-r})$$

Encrypt(I, M)

$$CT = (c_1, c_2, c_3, c_4, c_5)$$
$$= \left(M \oplus \mathsf{ext}\left(e(g,g)^{abz}, s\right), s, (u_0\prod_{i=1}^B u_i^{I_i})^z, g^z, e(g,g)^{xyz}\right)$$

Decrypt

$$\mathsf{ext}\left(e(s_2,c_4)e(s_3,c_3)c_5^{s_1}, c_2\right) \oplus c_1$$

4.2 Security

The proof follows the security proof of [16] modified to our IBE system and security game. In order to do that we need to define three additional games. We denote by $L = L(n)$ the maximum number of "leaked" identities, i.e. the attacker makes leakage queries for these identities. Also we denote by $K = K(n)$ the maximum number of KeyGen queries. Since each game is played against a polynomial-time attacker both L, K are polynomials in n. The total number of identities leaked and the number of Keygen queries by the attacker except the challenge identity is denoted by $Q = Q(n) = L(n) + K(n) - 1$. The games used are the following:

CpaLeak This is the regular secular security game as defined in 2.4. We will bound the advantage of any attacker on this game.

CpaMid This game is the same as the previous one with the exception that the challenger keeps track of all identities leaked and queried by the attacker in Phases 1 and 2. At the Guess phase he has acquired a vector $\vec{I} = (I^{(1)}, I^{(2)}, \cdots, I^{(Q)}, I^*)$ where I^* is the Challenge identity. In this phase the challenger sets $m = 2Q$, picks $B+1$ random elements $\vec{x} = (x_0, x_1, \cdots, x_B) \xleftarrow{R} [m-1]_0^{B+1}$, and a parameter $k \xleftarrow{R} [B]_0$.

We define the binary function

$$K(I) = \begin{cases} 0 & \text{if } x_0 + \sum_{i=1}^B x_i I_i = 0 \bmod m \\ 1 & \text{otherwise} \end{cases}$$

and the regular abort indicator function

$$\tau(\vec{I}, \vec{x}, k) = \begin{cases} 0 & \text{if } \left(\bigwedge_{i=1}^Q K(I^{(i)}) = 1\right) \wedge \\ & \quad \left(x_0 + \sum_{i=1}^B x_i I_i^* = km\right) \\ 1 & \text{otherwise} \end{cases}$$

Then the challenger evaluates $\tau(\vec{I}, \vec{x}, k)$ a total number of T times with fresh values of $\vec{x}, k$ in order to get an estimate ζ' of the probability $\zeta(\vec{I}) \equiv \Pr_{\vec{x},k}[\tau(\vec{I}, \vec{x}, k) = 0]$, i.e. $T \cdot \zeta'$ is the number of all tries where $\tau(\vec{I}, \vec{x}, k) = 0$. The parameter T will be defined later in the proof; here we assume it is a fixed parameter of all games.

The game is finished in one of three ways:

1. Regular Abort: If $\tau(\vec{I}, \vec{x}, k) = 1$ then the challenger flips a coin $\varphi \xleftarrow{R} \{0,1\}$ and the attacker succeeds in the game if $\varphi = 0$.

2. Artificial Abort: If $\zeta' \geq \zeta_m = 1/(4Q(B+1))$ the challenger aborts with probability $(\zeta' - \zeta_m)/\zeta'$. By abort we mean that he flips a coin $\varphi \stackrel{R}{\leftarrow} \{0,1\}$ and the attacker succeeds if $\varphi = 0$.

3. Final Phase: If none of the above events happened, the attacker wins if he correctly guessed the challenge bit σ chosen by the challenger in the Challenge phase (see definition 2.4).

CpaSim This game is the same as the previous game with the exception that the challenger tries to catch the regular aborts "on the fly". That is, if he is queried for an identity $I \neq I^*$ such that $K(I) = 0$ or for I^* if $x_0 + \sum_{i=1}^{B} x_i I_i^* \neq km$, he immediately aborts.

The crucial difference is that in order to do that he has to know before the Challenge phase which one is the challenge identity. To do that he picks a random integer $i^* \stackrel{R}{\leftarrow} [L]$ in the beginning and assumes that the i^*-th leaked identity is the Challenge one. If his guess is wrong, he aborts. This abort, called Challenge Abort, can either happen in Phase 1 when the attacker makes a Reveal or a KeyGen query on the i^*-th identity or in the Challenge phase if I^* is not the i^*-th identity. This guess will impose an $1/L$ factor in the resulting probability.

In all aborts (Regular, Artificial, or Challenge), he flips a coin $\varphi \stackrel{R}{\leftarrow} \{0,1\}$ and the attacker succeeds if $\varphi = 0$. Otherwise the attacker wins if his response $\sigma' = \sigma$.

CpaModfd The final game is the same as the previous one with the exception that in the Challenge phase the challenger returns the "modified" ciphertext:

$$
\begin{array}{ll}
z, w \stackrel{R}{\leftarrow} \mathbb{Z}_p & \sigma \stackrel{R}{\leftarrow} \{0,1\} \\
C = e(g,g)^{abz} e(g,g)^{xy(w-z)t^*} & c_2 \stackrel{R}{\leftarrow} \{0,1\}^{\mu} \\
c_1 = M_\sigma \oplus \mathsf{ext}(C, c_2) & c_3 = (u_0 \prod_{i=1}^{B} u_i^{I_i^*})^z \\
c_4 = g^z & c_5 = e(g,g)^{\frac{}{xyw}}
\end{array}
$$

where t^* is the tag of the secret key SK_{I^*} of the challenge identity leaked to the adversary during Phase 1.

In order to prove lemma 4.4 that relates the first two games, we need the following three claims. The proofs are the same as the ones in [25, 16]. Remember that $\zeta_m = \frac{1}{4Q(B+1)}$ in all claims.

Claim 4.1. *For any vector of $Q+1$ identities $\vec{I} = (I^{(1)}, I^{(2)}, \cdots, I^{(Q)}, I^*)$, where I^* is not equal to any $I^{(i)}$, we have that*

$$\zeta(\vec{I}) \equiv \Pr_{\vec{x},k}[\tau(\vec{I}, \vec{x}, k) = 0] \geq \zeta_m$$

Proof. For $\vec{I}$ we have that:

$$
\begin{aligned}
&\Pr_{\vec{x},k}[\tau(\vec{I}, \vec{x}, k) = 0] = \\
&= \Pr_{\vec{x},k}[\bigwedge_{i=1}^{Q} K(I^{(i)}) = 1 \wedge x_0 + \sum_{i=1}^{B} x_i I_i^* = km] = \\
&= \frac{1}{B+1} \Pr_{\vec{x}}[\bigwedge_{i=1}^{Q} K(I^{(i)}) = 1 \wedge K(I^*) = 0] = \qquad (1) \\
&= \frac{1}{B+1} \Pr_{\vec{x}}[K(I^*) = 0] \Pr_{\vec{x}}[\bigwedge_{i=1}^{Q} K(I^{(i)}) = 1 | K(I^*) = 0] = \\
&= \frac{1}{(B+1)m} \left(1 - \Pr_{\vec{x}}[\bigvee_{i=1}^{Q} K(I^{(i)}) = 0 | K(I^*) = 0]\right) \geq \qquad (2) \\
&\geq \frac{1}{(B+1)m} \left(1 - \sum_{i=1}^{Q} \Pr_{\vec{x}}[K(I^{(i)}) = 0 | K(I^*) = 0]\right) = \qquad (3) \\
&= \frac{1}{(B+1)m} \left(1 - \frac{Q}{m}\right) = \qquad (4) \\
&= \frac{1}{4Q(B+1)} \qquad (5)
\end{aligned}
$$

Equation 1 follows from the fact that k is chosen randomly among $B+1$ values. Equation 2 follows from $\Pr[K(I) = 0] = 1/m$ for any I. Inequality 3 is the Union Bound. Equation 4 follows from the pairwise independence of $K(I^{(i)}) = 0$ and $K(I^*) = 0$, since all $I^{(i)}$'s are different from I^* in at least one bit. Finally equation 5 is the result of substituting the optimal value $m = 2Q$. □

Claim 4.2. *If $T = 192\varepsilon^{-2}\zeta_m^{-1}\ln(8\zeta_m^{-1}\varepsilon^{-1})$ where $0 \leq \varepsilon \leq 1/2$ then for any vector of $Q+1$ identities $\vec{I} = (I^{(1)}, I^{(2)}, \cdots, I^{(Q)}, I^*)$, where I^* is not equal to any $I^{(i)}$, the probability over $\vec{x}, k$ that there is an abort (regular or artificial) in game* CpaMid *is at least $1 - \zeta_m - 3\zeta_m\varepsilon/8$.* [4]

Proof. We denote by ζ' the estimate of the probability $\zeta = \zeta(\vec{I})$ after T tries. Then by Chernoff Bounds [3] we have that:

$$
\begin{aligned}
&\Pr[T\zeta' \leq T\zeta\,(1 - \varepsilon/8)] \leq \\
&\quad \exp\left(-192\varepsilon^{-2}\zeta_m^{-1}\ln(8\zeta_m^{-1}\varepsilon^{-1})\zeta(\varepsilon/8)^2/2\right) \Rightarrow \\
&\Pr[\zeta' \leq \zeta\,(1 - \varepsilon/8)] \leq \zeta_m\varepsilon/8
\end{aligned}
$$

where we used the fact that $\zeta \geq \zeta_m$ by claim 4.1.

An artificial abort will not occur with probability ζ_m/ζ'. Thus if we denote by AA the event of an artificial abort, by RA a regular abort, and by X the event that $\zeta' > \zeta(1-\varepsilon/8)$

[4] If ε is negligible in n this makes T super-polynomial and our reductions will not work. However as we will see, we will set $\varepsilon = \mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},P}$. Thus if ε is negligible, we know immediately that our system is secure.

we have that:

$$\begin{aligned}
&\Pr[abort] = \\
&\quad = 1 - \Pr[\overline{abort}] = 1 - \Pr[\overline{RA}]\Pr[\overline{AA}] = \\
&\quad = 1 - \Pr[\overline{RA}]\left(\Pr[\overline{AA}|X]\Pr[X] + \Pr[\overline{AA}|\overline{X}]\Pr[\overline{X}]\right) \geq \\
&\quad = 1 - \Pr[\overline{RA}]\left(\Pr[\overline{AA}|X] + \Pr[\overline{X}]\right) \geq \\
&\quad \geq 1 - \zeta\left(\frac{\zeta_m}{\zeta(1-\varepsilon/8)} + \frac{\zeta_m\varepsilon}{8}\right) \geq \\
&\quad \geq 1 - \left(\frac{\zeta_m}{1-\varepsilon/8} + \frac{\zeta_m\varepsilon}{8}\right) \geq \\
&\quad \geq 1 - \left(\zeta_m\left(1 + \frac{2\varepsilon}{8}\right) + \frac{\zeta_m\varepsilon}{8}\right) \geq \\
&\quad \geq 1 - \zeta_m - \frac{3\zeta_m}{8}
\end{aligned}$$

□

CLAIM 4.3. *If $T = 192\varepsilon^{-2}\zeta_m^{-1}\ln(8\zeta_m^{-1}\varepsilon^{-1})$ where $0 \leq \varepsilon \leq 1/2$ then for any vector of $Q+1$ identities $\vec{I} = (I^{(1)}, I^{(2)}, \cdots, I^{(Q)}, I^*)$, where I^* is not equal to any $I^{(i)}$, the probability over $\vec{x}, k$ that there is no abort (regular or artificial) in game* CpaMid *is at least $\zeta_m - \zeta_m\varepsilon/4$.*

PROOF. By applying Chernoff Bounds as before [3], we have:

$$\begin{aligned}
&\Pr[T\zeta' \geq T\zeta(1+\varepsilon/8)] \leq \\
&\quad \exp\left(-192\varepsilon^{-2}\zeta_m^{-1}\ln(8\zeta_m^{-1}\varepsilon^{-1})\zeta(\varepsilon/8)^2/3\right) \Rightarrow \\
&\Pr[\zeta' \geq \zeta(1+\varepsilon/8)] \leq \zeta_m\varepsilon/8
\end{aligned}$$

If we denote by Y the event that $\zeta' \geq \zeta(1+\varepsilon/8)$ we have that:

$$\begin{aligned}
\Pr[\overline{abort}] &= \Pr[\overline{RA}]\Pr[\overline{AA}] \geq \\
&\geq \Pr[\overline{RA}]\Pr[\overline{AA}|\overline{Y}](1 - \Pr[Y]) \geq \\
&\geq \zeta \cdot \frac{\zeta_m}{\zeta(1+\varepsilon/8)} \cdot \left(1 - \frac{\zeta_m\varepsilon}{8}\right) \geq \\
&\geq \zeta_m\left(1 - \frac{\varepsilon}{4}\right)
\end{aligned}$$

□

LEMMA 4.4. *For any algorithm $\mathcal{A}$ it is true that*

$$\mathsf{Adv}^{\mathsf{CpaMid}}_{\mathcal{A},P} \geq \frac{9\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},P}}{64Q(B+1)}$$

where $Q = L + K - 1$ is the number of leaked identities and keygen queries minus 1, and B is the number of bits of every identity.

PROOF. If we denote by A the event that an abort occurs in CpaMid, by $S1$ the event that $\mathcal{A}$ succeeds in game CpaLeak, and by $S2$ that he succeeds in game CpaMid, we get that:

$$\begin{aligned}
\Pr[S2] &= \Pr[S2|A]\Pr[A] + \Pr[S2|\overline{A}]\Pr[\overline{A}] \\
&= \Pr[S2|A]\Pr[A] + \Pr[S1|\overline{A}]\Pr[\overline{A}] && (6)\\
&= \Pr[S2|A]\cdot\Pr[A] + \Pr[S1]\Pr[\overline{A}|S1] && (7)\\
&= \frac{1}{2}\cdot\Pr[A] + \left(\frac{1}{2} + \mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},P}\right)\Pr[\overline{A}|S1] && (8)\\
&\geq \frac{1}{2}\cdot(1 - \zeta_m - 3\zeta_m\varepsilon/8) + \\
&\quad + \left(\frac{1}{2} + \mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},P}\right)(\zeta_m - \zeta_m\varepsilon/4) && (9)
\end{aligned}$$

Equation 6 follows from the observation that when no abort occurs the two games are identical. Equation 7 is an application of Bayes' Theorem. Equation 8 follows from the fact that the probability of success given an abort is exactly $1/2$ and the definition of advantage in game CpaLeak. Equation 9 follows from claims 4.2 and 4.3. We got rid of the conditional dependence on $S1$, since according to lemma 4.3 for any valid vector of identities the probability of abort depends only on the choices of $\vec{x}, k$ not used in game CpaLeak.

If we set $\varepsilon = \mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},P}$ and $\Pr[S2] = 1/2 + \mathsf{Adv}^{\mathsf{CpaMid}}_{\mathcal{A},P}$ we get that:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{CpaMid}}_{\mathcal{A},P} &\geq -\frac{\zeta_m}{2} - \frac{3\zeta_m\varepsilon}{16} + \frac{\zeta_m}{2} - \frac{\zeta_m\varepsilon}{8} + \zeta_m\varepsilon - \frac{\zeta_m\varepsilon^2}{4} \\
&\geq \frac{11\zeta_m\varepsilon}{16} - \frac{\zeta_m\varepsilon}{8} \\
&\geq \frac{9\varepsilon}{64Q(B+1)}
\end{aligned}$$

The second inequality is due to the fact that $\varepsilon \leq 1/2$ and the last one if we set $\zeta_m = 1/(4Q(B+1))$. □

LEMMA 4.5. *For any algorithm $\mathcal{A}$ it is true that*

$$\mathsf{Adv}^{\mathsf{CpaSim}}_{\mathcal{A},P} = \frac{\mathsf{Adv}^{\mathsf{CpaMid}}_{\mathcal{A},P}}{L}$$

where L is the number of the leakage queries.

PROOF. We observe that when a Challenge Abort does not occur in the CpaSim game then the two games are equivalent with respect to the success of the adversary. The only thing that changes is the time at which the regular aborts occur. The artificial aborts occur at the same time; at the end of the games. The regular aborts either occur "on the fly" for game CpaSim or at the end for game CpaMid. The result is that the success of $\mathcal{A}$ is random in both games. All parameters and computations have exactly the same distributions up to the point of the possible regular abort.

Therefore if we denote by $\overline{ChA}$ the event that a Challenge Abort does not occur in the CpaSim game, i.e. that the challenger guessed correctly I^*, and by S the event that $\mathcal{A}$ succeeds in this game, we get that:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{CpaSim}}_{\mathcal{A},P} &= \Pr[S] - \frac{1}{2} \\
&= \Pr[S|ChA]\Pr[ChA] + \\
&\quad + \Pr[S|\overline{ChA}]\Pr[\overline{ChA}] - \frac{1}{2} \\
&= \frac{1}{2}\left(1 - \frac{1}{L}\right) + \left(\frac{1}{2} + \mathsf{Adv}^{\mathsf{CpaMid}}_{\mathcal{A},P}\right)\frac{1}{L} - \frac{1}{2} \\
&= \frac{\mathsf{Adv}^{\mathsf{CpaMid}}_{\mathcal{A},P}}{L}
\end{aligned}$$

where the probabilities in the third equation follow from the facts that given a Challenge Abort the success probability of $\mathcal{A}$ is $1/2$, the probability of a Challenge Abort is $1-1/L$, and according to the previous argument, given that we don't challenge-abort, the success probability is the same as the success probability of the CpaMid game. $\square$

LEMMA 4.6. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\mathsf{CpaSim}}_{\mathcal{A},P} - \mathsf{Adv}^{\mathsf{CpaModfd}}_{\mathcal{A},P} = \epsilon$$

Then we can build a PPT algorithm $\mathcal{B}$ with advantage $\epsilon/2$ in breaking the DBDH assumption.

PROOF. The simulator $\mathcal{B}$ sets a parameter $m = 2Q$ during the Setup phase of the security game and picks a random integer $k \xleftarrow{R} [B]_0$. Remember that $Q = L+K-1$ and B is the number of bits of each identity. All of them are polynomial parameters in n. He picks $B+1$ random elements $\vec{x} = (x_0, x_1, \cdots, x_B) \xleftarrow{R} [m-1]_0^{B+1}$ and $B+1$ random elements $\vec{y} = (y_0, y_1, \cdots, y_B) \xleftarrow{R} \mathbb{Z}_p^{B+1}$.

$\mathcal{B}$ takes as input a DBDH challenge g, g^x, g^y, g^z, T_ν where $T_\nu \in \{e(g,g)^{xyz}, e(g,g)^{xyw}\}$. He sets implicitly $a = xt^* + \tilde{a}$ and $b = y + \tilde{b}$ as in the proof of lemma 3.1. Similar to that he can compute the public parameters $e(g,g)^{xy}$ and $e(g,g)^{ab}$. For the u_i's he sets $u_0 = (g^x)^{p-mk+x_0} g^{y_0}$ and $u_i = (g^x)^{x_i} g^{y_i}$. Since y_i's are random in $\mathbb{Z}_p$ all parameters are properly distributed and $\mathcal{B}$ sends the public parameters to $\mathcal{A}$.

For Phase 1 $\mathcal{B}$ picks a random integer $i^* \xleftarrow{R} [L]$ as a guess for the challenge identity. He will treat the secret key of the i^*-th identity differently than the rest. For ease of the analysis we define the following functions on identities $I \in \{0,1\}^B$:

$$F(I) = p - mk + x_0 + \sum_{i=1}^{B} x_i I_i \qquad J(I) = y_0 + \sum_{i=1}^{B} y_i I_i$$

Whenever $\mathcal{A}$ makes a leakage or a keygen query on an identity I other than the i^*-th leaked identity, $\mathcal{B}$ picks $\tilde{t}, \tilde{r} \xleftarrow{R} \mathbb{Z}_p$ and sets implicitly $r = -y\tilde{t}/F(I) + \tilde{r}$. He responds with the following secret key:

$$\begin{aligned}
s'_1 &= t^* - \tilde{t} \\
s'_2 &= g^{ab} g^{-xyt} (u_0 \prod_{i=1}^{k} u_i^{I_i})^r = \\
&= g^{xyt^*} g^{xt^*\tilde{b}} g^{y\tilde{a}} g^{\tilde{a}\tilde{b}} \cdot g^{-xyt^*} g^{xy\tilde{t}} \cdot (g^x)^{F(I)r} g^{J(I)r} = \\
&= (g^x)^{t^*\tilde{b}} (g^y)^{\tilde{a}} g^{\tilde{a}\tilde{b}} (g^x)^{\tilde{r}F(I)} (g^y)^{-\tilde{t}J(I)/F(I)} g^{\tilde{r}J(I)} \\
s'_3 &= r^{-r} = (g^y)^{\tilde{t}/F(I)} g^{-\tilde{r}}
\end{aligned}$$

However the simulator can not create the above keys if $F(I) = 0 \bmod p$. We require that when $K(I) = 0$ then the simulator aborts and responds to the DBDH challenger with a random bit $\nu' \xleftarrow{R} \{0,1\}$. If $K(I) \neq 0$ then $x_0 + \sum_{i=1}^{B} x_i I_i = mk' + r_m$, where $0 \leq k' \leq B$ and $1 \leq r_m \leq m-1$. Thus $F(I) = p + (k'-k)m + r_m$. Since $p = \Theta(2^n)$ and B, m are polynomials in n we have that $p >> (k'-k)m + r_m$ for infinitely many n's. Therefore $K(I) \neq 0$ is a sufficient condition for $F(I) \neq 0 \bmod p$.

On the other hand, when $\mathcal{A}$ asks for a leakage on the i^*-th identity I', we require that $x_0 + \sum_{i=1}^{B} x_i I'_i = mk$. Otherwise $\mathcal{B}$ quits the game and responds with a random bit. As we will see $\mathcal{B}$ uses this restriction in the Challenge phase. In Phase 1 he creates the following secret key $SK_{I'}$ in order to answer the leakage queries:

$$\begin{aligned}
s_1^* &= t^* & s_3^* &= g^{-r} \\
s_2^* &= (g^x)^{t^*\tilde{b}} (g^y)^{\tilde{a}} g^{\tilde{a}\tilde{b}} \cdot (u_0 \textstyle\prod_{i=1}^{B} u_i^{I'_i})^r
\end{aligned}$$

where $r \xleftarrow{R} \mathbb{Z}_p$.

In the Challenge phase $\mathcal{A}$ responds with an identity I^* and two messages M_0, M_1. If I^* is not the i^*-th leaked identity, $\mathcal{B}$ aborts and outputs a random answer $\nu' \xleftarrow{R} \{0,1\}$. Otherwise we have that $x_0 + \sum_{i=1}^{B} x_i I_i^* = mk$ and therefore $F(I^*) = 0 \bmod p$. $\mathcal{B}$ flips a random coin $\sigma \xleftarrow{R} \{0,1\}$ and returns the ciphertext:

$$\begin{aligned}
c_2^* &\xleftarrow{R} \{0,1\}^\mu & c_3^* &= (u_0 \textstyle\prod_{i=1}^{B} u_i^{I_i^*})^z = (g^z)^{J(I^*)} \\
c_4^* &= g^z & c_5^* &= T_\nu \\
C &= e(s_2^*, c_4^*) e(s_3^*, c_3^*) (c_5^*)^{s_1^*} & c_1^* &= M_\sigma \oplus \mathsf{ext}(C, c_2^*)
\end{aligned}$$

For Phase 2 $\mathcal{B}$ calculates the requested secret keys as in Phase 1. After $\mathcal{A}$'s response with σ', $\mathcal{B}$ executes the estimation of ζ' by computing T times the regular abort indicator function with fresh values $\vec{x}, k$. If the artificial abort conditions hold, he aborts and returns a random answer to the DBDH challenger.

Otherwise he returns $\nu' = 0$ if $\sigma' = \sigma$ and 1 otherwise.

Notice that $\mathcal{B}$ returns 0 either when he aborts and the random answer is 0 or when $\mathcal{A}$ succeeds. According to the definition of our games this is exactly the same as saying that $\mathcal{B}$ returns 0 when $\mathcal{A}$ succeeds, since in case of an abort the success of $\mathcal{A}$ is determined by a random bit.

The rest of the proof is exactly the same as the final arguments of lemma's 3.1 proof:

$$\begin{aligned}
&\frac{1}{2}\left(\Pr[\mathcal{B}(D, e(g,g)^{xyz}) = 0] - \Pr[\mathcal{B}(D, e(g,g)^{xyw}) = 0]\right) = \\
&\frac{1}{2}\left(\frac{1}{2} + \mathsf{Adv}^{\mathsf{CpaSim}}_{\mathcal{A},P} - \left(\frac{1}{2} + \mathsf{Adv}^{\mathsf{CpaModfd}}_{\mathcal{A},P}\right)\right) \\
&\quad = \frac{1}{2}\left(\mathsf{Adv}^{\mathsf{CpaSim}}_{\mathcal{A},P} - \mathsf{Adv}^{\mathsf{CpaModfd}}_{\mathcal{A},P}\right) = \frac{\epsilon}{2}
\end{aligned}$$

where $D = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, g^y, g^z)$. $\square$

The following lemma is similar to lemma 3.2 and the proof is exactly the same:

LEMMA 4.7. *For any PPT adversary $\mathcal{A}$ it holds that*

$$\mathsf{Adv}^{\mathsf{CpaModfd}}_{\mathcal{A},P} \leq 2\epsilon_{\mathsf{ext}}$$

THEOREM 4.8. *If the DBDH assumption holds and the extractor's second parameter ϵ_{ext} used in P is negligible in n, then system P is l-leakage fully secure, where $l = \log|\mathbb{G}_T| - k$ and k is the extractor's first parameter.*

PROOF. By lemmas 4.4 and 4.5 we have that

$$\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},P} \leq \frac{64}{9} LQ(B+1) \mathsf{Adv}^{\mathsf{CpaSim}}_{\mathcal{A},P}$$

If we denote by ϵ_{DBDH} the maximum advantage of any PPT attacker of the DBDH game, by lemmas 4.6 and 4.7

we get that:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},P} &\leq \frac{64}{9}LQ(B+1)\left(2\epsilon_{DBDH}+\mathsf{Adv}^{\mathsf{CpaModfd}}_{\mathcal{A},P}\right)\\ &\leq \frac{128}{9}LQ(B+1)\left(\epsilon_{DBDH}+\epsilon_{\mathsf{ext}}\right)\end{aligned}$$

Since we care about PPT attackers, L, Q are polynomials in n. This is also true for B. Therefore if the premises of the theorem are true, the advantage of any PPT attacker is negligible. □

Leakage Summary The total amount of leakage is slightly less than 1/3 of the secret key for the same reasons as in system Π.

5. OUR THIRD SYSTEM

Our system is based on the Dual Encryption system of Lewko - Waters [20], designed in order to achieve full security with smaller public parameters' size. The transformation we apply is similar to the one we applied in Boneh-Boyen system, where the parameters instead of $a \cdot b$ and $x \cdot y$ are α and β respectively. However now the security proof is much more complicated since we have to move from the original security game to a game where all secret keys and the ciphertext have a specific form called semi-functional.

5.1 Construction

The following algorithms consist our IBE system, denoted Σ:

Setup The setup algorithm uses a group generator $\mathcal{G}(1^n)$ to create a bilinear group $\mathbb{G}$ of composite order $N = p_1p_2p_3$. It then picks uniformly at random $\alpha, \beta \overset{R}{\leftarrow} \mathbb{Z}_N$ and $g, u, h \overset{R}{\leftarrow} \mathbb{G}_{p_1}$. It also chooses an extractor function ext with parameters $(\log|\mathbb{G}_{T,1}| - l, \epsilon_{\mathsf{ext}})$, where $\mathbb{G}_{T,1}$ is the subgroup of $\mathbb{G}_T$ of order p_1.

Finally it publishes the public parameters

$$PP = (N, \mathbb{G}, \mathbb{G}_T, e, g, u, h, e(g,g)^{\alpha}, e(g,g)^{\beta}, \mathsf{ext})$$

and stores privately the master secret key $MSK = (\alpha, \beta, g_{p_3})$ where g_{p_3} is a generator of $\mathbb{G}_{p_3}$.

KeyGen(MSK, I) The key generation algorithm initially picks two exponents $t, r \overset{R}{\leftarrow} \mathbb{Z}_N$ and two exponents $\rho, \rho' \overset{R}{\leftarrow} \mathbb{Z}_N$.

Then the algorithm calculates the following secret key:

$$SK = (s_1, s_2, s_3) = (t, g^{\alpha}g^{-\beta t}(u^I h)^r g_{p_3}^{\rho}, g^{-r}g_{p_3}^{\rho'})$$

Encrypt(I, M) The encryption algorithm picks an exponent $z \overset{R}{\leftarrow} \mathbb{Z}_N$ and a random seed $s \in \{0,1\}^{\mu}$ for the extractor function. The ciphertext is:

$$\begin{aligned}CT &= (c_1, c_2, c_3, c_4, c_5)\\ &= \left(M \oplus \mathsf{ext}\left(e(g,g)^{\alpha z}, s\right), s, (u^I h)^z, g^z, e(g,g)^{\beta z}\right)\end{aligned}$$

Decrypt The decryption algorithm is

$$\mathsf{ext}(e(s_2, c_4)e(s_3, c_3)c_5^{s_1}, c_2) \oplus c_1$$

If the identities of the secret key and the ciphertext are equal, the decryption algorithm works as follows:

$$\begin{aligned}&c_1 \oplus \mathsf{ext}(e(s_2, c_4)e(s_3, c_3)c_5^{s_1}, c_2)\\ &\quad = M \oplus \mathsf{ext}\left(e(g,g)^{\alpha z}, s\right) \oplus\\ &\quad \oplus \mathsf{ext}(e(g,g)^{\alpha z}e(g,g)^{-\beta tz}e(u^I h, g)^{rz} \cdot\\ &\quad \cdot e(g_{p_3}, g)^{z\rho}e(u^I h, g)^{-rz}e(u^I h, g_{p_3})^{z\rho'}e(g,g)^{\beta tz})\\ &\quad = M\end{aligned}$$

5.2 Security

To prove leakage-resilient security of our scheme we will use the additional structures, defined in [20], of semi - functional ciphertexts and semi-functional keys. The ciphertexts and keys generated by **KeyGen** and **Encrypt** will be referred to as normal. In the following definitions we let g_{p_2} denote a generator of the subgroup $\mathbb{G}_{p_2}$.

DEFINITION 5.1. *To create a semi-functional key, firstly a normal key (s_1, s_2, s_3) is created. Then two random exponents $\gamma, z_k \overset{R}{\leftarrow} \mathbb{Z}_N$ are chosen. The semi-functional key is $(s_1, s_2 g_{p_2}^{\gamma}, s_3 g_{p_2}^{\gamma z_k})$.*

DEFINITION 5.2. *To create a semi-functional ciphertext, firstly a normal ciphertext $(c_1, c_2, c_3, c_4, c_5)$ is created. Then two random exponents $\delta, z_c \overset{R}{\leftarrow} \mathbb{Z}_N$ are chosen. The semi-functional ciphertext is $(c_1, c_2, c_3 g_{p_2}^{\delta z_c}, c_4 g_{p_2}^{\delta}, c_5)$.*

The proof of security relies on Assumptions 1,2,3 of Section 2.6.2. We will define a sequence of games and use them in a hybrid proof of security.

$\mathsf{CpaLeak}$ The first game is the normal security game as defined in 2.4.

$\mathsf{Restricted}$ The next game is the same except that all identities the attacker queries are not equal to the challenge identity $I^* \bmod p_2$.[5] That means that when the attacker gives an I^* such that for some I in Phase 1, $I^* = I \bmod p_2$ or when this is the case in Phase 2, the challenger picks a random bit $\varphi \overset{R}{\leftarrow} \{0,1\}$ and the attacker succeeds if $\varphi = 0$. We will retain this restriction in all subsequent games.

Leak_i We denote by $L(n)$ the maximum number of different identities used in leakage queries. Then for each $i \in [L-1]_0$ we define the game Leak_i to be like the $\mathsf{Restricted}$ security game but the ciphertext is semi-functional and the leaked keys of the first i identities are semi-functional excluding the key of the challenge identity. In game Leak_0 all keys are normal and the ciphertext is semi-functional. In game Leak_{L-1} all leaked keys except the key of the challenge identity are semi-functional. The keys of KeyGen queries are all normal. Remember that we assume that the attacker always makes a leakage query on the challenge identity; hence the total number of identities leaked if we don't include the challenge identity is $L-1$.

KG_i We denote by $K(n)$ the maximum number of KeyGen queries and define games KG_i for $1 \leq i \leq K$ as follows: In game KG_i the ciphertext is semi-functional, all leaked keys but the one of the challenge identity are semi-functional, and the first i keys generated by KeyGen queries are semi-functional. The remaining keys are all normal. Notice that $K(n)$ is the number of queries and not the number of different identities as in $L(n)$. That is why we treat them differently.

[5] Obviously we exclude I^*.

Final The difference of this game from game KG_K is that the ciphertext is a "modified" semi-functional ciphertext. By "modified" we mean

$$\begin{array}{ll} z, w \xleftarrow{R} \mathbb{Z}_N & \sigma \xleftarrow{R} \{0,1\} \\ C = e(g,g)^{\alpha z} e(g,g)^{\beta(w-z)t^*} & c_2 \xleftarrow{R} \{0,1\}^\mu \\ c_1 = M_\sigma \oplus \mathsf{ext}(C, c_2) & c_3 = (u^{I^*} h)^z g_{p_2}^{\delta z_c} \\ c_4 = g^z g_{p_2}^{\delta} & c_5 = e(g,g)^{\beta w} \end{array}$$

where t^* is the tag used in the key of the challenge identity.

The advantages of all algorithms in the above games are defined in the same way as $\mathsf{Adv}^{\mathsf{CpaLeak}}$. In the following lemmas we will prove that all of these games are indistinguishable by PPT attackers if the assumptions in 2.6.2 are true.

LEMMA 5.3. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Restricted}}_{\mathcal{A},\Sigma} = \epsilon$. Then we can build a PPT algorithm $\mathcal{B}$ with advantage at least $\epsilon/4$ in breaking either Assumption 1 or Assumption 2.*

PROOF. In both assumptions simulator $\mathcal{B}$ is given g_{p_1} and g_{p_3}. Therefore $\mathcal{B}$ can simulate a "full" version of Σ by picking uniformly at random exponents $x, y, \alpha, \beta \xleftarrow{R} \mathbb{Z}_N$ and setting $g = g_{p_1}, u = g^x, h = g^y$. Since he knows the master secret key (α, β, g_{p_3}), he can generate secret keys for all identities, answer to all leakage queries, and encrypt any message in the challenge phase.

According to the lemma's assumption, $\mathcal{A}$ will query for identities $I \neq I^* \bmod N$ and $I = I^* \bmod p_2$ with probability ϵ. This means that in the end $\mathcal{B}$ can compute a non-trivial factor of $N = p_1 p_2 p_3$ by calculating $a = \gcd(I - I^*, N)$. If he computes $b = N/a$, it is true that with probability ϵ:

$$\begin{array}{c} \textit{Case 1: } b = p_1 \text{ or } b = p_1 p_3 \\ \textit{Case 2: } b = p_3 \end{array}$$

One of the two cases occurs with probability at least $\epsilon/2$. In case 1, $\mathcal{B}$ breaks assumption 1 by raising the challenge term T_ν of the assumption to b. If $\nu = 1$ then $T_\nu^b = 1$. Otherwise $T_\nu \neq 1$.

In case 2, $\mathcal{B}$ breaks assumption 2 by testing if

$$e\left((g_{p_2}^{x_2} g_{p_3}^{x_3})^b, T_\nu\right) = 1$$

Notice that $b = p_3$ and thus the first term in the pairing is $g_{p_2}^{x_2 p_3}$. If T_ν contains no p_2 part, i.e. when $\nu = 1$, the above pairing equals 1. Otherwise $T_\nu \in \mathbb{G}$ and $\nu = 0$. □

LEMMA 5.4. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{Restricted}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Leak}_0}_{\mathcal{A},\Sigma} = \epsilon$. Then we can build a PPT algorithm $\mathcal{B}$ with advantage $\epsilon/2$ in breaking Assumption 1.*

PROOF. As before $\mathcal{B}$ is given g_{p_1} and g_{p_3} and he can simulate fully the encryption system Σ. He sets $g = g_{p_1}, u = g^x, h = g^y$. $\mathcal{B}$ can answer all leakage and keygen queries using the master secret key (α, β, g_{p_3}).

In the Challenge phase $\mathcal{A}$ sends two messages M_0, M_1 and the challenge identity I^*. $\mathcal{B}$ picks $\sigma \xleftarrow{R} \{0,1\}$ and responds with the following ciphertext using the challenge term T_ν:

$$\begin{array}{ll} c_2^* \xleftarrow{R} \{0,1\}^\mu & c_3^* = T_\nu^{xI^*+y} \\ c_4^* = T_\nu & c_5^* = e(T_\nu, g)^\beta = e(g,g)^{\alpha\beta} \\ C = e(T_\nu, g)^\alpha = e(g,g)^{\alpha\alpha} & c_1^* = M_\sigma \oplus \mathsf{ext}(C, c_2^*) \end{array}$$

If $T_\nu = g_{p_1}^a$, we get a normal ciphertext and $\mathcal{A}$ plays game $\mathsf{Restricted}$ (where $z = a$). If $T_\nu = g_{p_1}^a g_{p_2}^b$, then the above is a semi-functional ciphertext with $z_c = xI^* + y$ and $\mathcal{A}$ plays game Leak_0. The value of z_c modulo p_2 is not correlated with the values of x and y modulo p_1 so this is correctly distributed. Hence, if $\mathcal{B}$ answers $\nu' = 0$ when $\mathcal{A}$ succeeds, he has advantage $\epsilon/2$ in breaking assumption 1. □

LEMMA 5.5. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{Leak}_{i-1}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Leak}_i}_{\mathcal{A},\Sigma} = \epsilon$. Then we can build a PPT algorithm $\mathcal{B}$ with advantage at least $\epsilon/(2L)$ in breaking Assumption 2.*

PROOF. Algorithm $\mathcal{B}$ has to create semi-functional keys for some identities excluding the challenge identity. Since he does not know I^* before the challenge phase, he picks uniformly at random a $i^* \xleftarrow{R} [L]$ as a guess for the challenge identity. With probability $1/L$, the guess is correct. Notice that according to the assumptions on how the adversaries work the challenge identity is in the list of the leaked identities. Therefore for the i^*-th leaked identity the secret key is normal, regardless of its position.

As before $\mathcal{B}$ picks random exponents $x, y, \alpha, \beta \xleftarrow{R} \mathbb{Z}_N$ and sets $g = g_{p_1}, u = g^x, h = g^y$. For Phase 1 queries he answers all Keygen queries and the i^*-th leaked identity query with normal keys using the $MSK = (\alpha, \beta, g_{p_3})$.

For the first $i - 1$ leakage[6] identities he responds with semi-functional keys. In order to do that he uses the input of assumption 2:

$$\begin{array}{c} s_1 = t \quad s_3 = g^{-r}(g_{p_2}^{x_2} g_{p_3}^{x_3})^{\rho'} g_{p_3}^{\rho''} \\ s_2 = g^\alpha g^{-\beta t}(u^I h)^r (g_{p_2}^{x_2} g_{p_3}^{x_3})^\rho \end{array}$$

where I is the queried identity and $t, r, \rho, \rho', \rho'' \xleftarrow{R} \mathbb{Z}_N$. The above is a properly distributed semi-functional key with $\gamma = x_2 \rho$ and $z_k = \rho' \rho^{-1} \bmod p_2$.

For the i-th leakage identity he uses the challenge of assumption 2 to create the key:

$$\begin{array}{ll} s_1 = t & s_3 = T_\nu \\ s_2 = g^\alpha g^{-\beta t} T_\nu^{z_k} g_{p_3}^\rho & \end{array}$$

where $t, \rho \xleftarrow{R} \mathbb{Z}_N$ and $z_k = xI + y$. For the remaining leakage queries $\mathcal{B}$ creates normal secret keys using his master secret key.

At the Challenge phase $\mathcal{A}$ responds with two messages M_0, M_1 and the challenge identity I^*. If $\mathcal{B}$ didn't make a correct guess for the challenge identity, he aborts at this point. Otherwise he flips a random coin $\sigma \xleftarrow{R} \{0,1\}$ and returns the following challenge ciphertext:

$$\begin{array}{ll} c_2^* \xleftarrow{R} \{0,1\}^\mu & c_3^* = (g_{p_1}^{x_1} g_{p_2})^{xI^*+y} \\ c_4^* = g_{p_1}^{x_1} g_{p_2} & c_5^* = e(g_{p_1}^{x_1} g_{p_2}, g)^\beta = \\ & \quad = e(g,g)^{\beta x_1} \\ C = e(g_{p_1}^{x_1} g_{p_2}, g)^\alpha = e(g,g)^{\alpha x_1} & c_1^* = M_\sigma \oplus \mathsf{ext}(C, c_2^*) \end{array}$$

This is a properly distributed semi-functional ciphertext with $z_c = xI^* + y$.

This is where we use our modular restriction that all identities are different to the challenge identity modulo p_2. Since for all queries $I \neq I^* \bmod p_2$, we have that the $z_k = xI + y$ for the i-th identity and the $z_c = xI^* + y$ seem randomly distributed to $\mathcal{A}$ modulo p_2. This relationship is the reason that we can not create a semi-functional secret key for the

[6]Remember that by leakage queries we refer to both $\mathsf{Leakage}$ and Reveal queries

challenge identity. Then we would have $z_c = z_k$ and obviously they wouldn't be properly distributed, i.e. random.

For Phase 2 $\mathcal{B}$ answers with normal secret keys.

Notice that if $T_\nu = g_{p_1}^a g_{p_3}^c$ then the secret key of the i-th leaked identity is normal. Thus $\mathcal{A}$ played game Leak_{i-1}. If $T_\nu = g_{p_1}^a g_{p_2}^b g_{p_3}^c$ then this secret key is semi-functional and $\mathcal{A}$ played the Leak_i game. Therefore if $\mathcal{B}$ answers $\nu' = 0$ when $\mathcal{A}$ succeeds, he breaks assumption 2 with probability $\epsilon/(2L)$. □

LEMMA 5.6. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{Leak}_{L-1}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{KG}_1}_{\mathcal{A},\Sigma} = \epsilon$. Then we can build a PPT algorithm $\mathcal{B}$ with advantage at least $\epsilon/(2L)$ in breaking Assumption 2.*

LEMMA 5.7. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{KG}_{i-1}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{KG}_i}_{\mathcal{A},\Sigma} = \epsilon$. Then we can build a PPT algorithm $\mathcal{B}$ with advantage $\epsilon/(2L)$ in breaking Assumption 2.*

PROOF. For lemmas 5.6 and 5.7 the proofs are similar to the proof of lemma 5.5. The simulator guesses the challenge identity with probability $1/L$ and creates semi-functional secret keys for all leaked identities except the guessed challenge identity.

Then he uses the challenge term of assumption 2 to create the secret key of the 1st or the i-th KeyGen query for the simulation of lemma 5.6 and 5.7, respectively. Keygen queries before the i-th one return semi-functional secret keys and after that normal secret keys. Note that in this case we count the queries and not the identities. Also the i-th query may lie in Phase 2; contrary to the leakage queries which are only made in Phase 1. □

LEMMA 5.8. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{KG}_K}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Final}}_{\mathcal{A},\Sigma} = \epsilon$. Then we can build a PPT algorithm $\mathcal{B}$ with advantage $\epsilon/(2L)$ in breaking Assumption 3.*

PROOF. According to Assumption 3 $\mathcal{B}$ first receives

$$(g_{p_1}, g_{p_3}, g_{p_1}^\beta g_{p_2}, g_{p_1}^z g_{p_2}^{x_2}, g_{p_2}^{y_2}, T_\nu)$$

It chooses random exponents $x, y, t^*, \tilde{\alpha} \xleftarrow{R} \mathbb{Z}_N$ and sets implicitly $\alpha = t^*\beta + \tilde{\alpha}$. Notice that he does not know the master secret key. He calculates the public parameters as:

$$g = g_{p_1} \qquad u = g_{p_1}^x \qquad h = g_{p_1}^y$$
$$e(g,g)^\beta = e(g_{p_1}^\beta g_{p_2}, g_{p_1}) \qquad e(g,g)^\alpha = \left(e(g,g)^\beta\right)^{t^*} e(g_{p_1}, g_{p_1})^{\tilde{\alpha}}$$

and sends them to $\mathcal{A}$. As in the other proofs $\mathcal{B}$ guesses the challenge identity by picking a random $i^* \in [L]$. With probability $1/L$ the guess is correct.

In Phase 1, when $\mathcal{A}$ asks for a key on identity $I \neq I^{(i^*)}$ either through a leakage or either through a KeyGen query, $\mathcal{B}$ generates the following semi-functional keys. It picks random exponents $\tilde{t}, r, \rho, \rho', \rho'', \rho''' \xleftarrow{R} \mathbb{Z}_N$ and computes:

$$s_1' = t^* - \tilde{t} \qquad s_3' = g_{p_1}^{-r}(g_{p_2}^{y_2})^{\rho''} g_{p_3}^{\rho'}$$
$$s_2' = (g_{p_1}^\beta g_{p_2})^{\tilde{t}} g_{p_1}^{\tilde{\alpha}} (u^I h)^r g_{p_3}^{\rho} (g_{p_2}^{y_2})^{\rho'''}$$

The above is a properly distributed semi-functional secret key, because $(g_{p_1}^\beta)^{\tilde{t}} g_{p_1}^{\tilde{\alpha}} = g_{p_1}^{\alpha} g_{p_1}^{-\beta s_1'}$.

For the secret key of $I^{(i^*)}$, $\mathcal{B}$ picks random $r, \rho, \rho' \xleftarrow{R} \mathbb{Z}_N$ and calculates the following:

$$s_1^* = t^*, \quad s_2^* = g_{p_1}^{\tilde{\alpha}} (u^{I^{(i^*)}} h)^r g_{p_3}^{\rho}, \quad s_3^* = g_{p_1}^{-r} g_{p_3}^{\rho'}$$

The above is a correctly distributed normal key for $I^{(i^*)}$.

$\mathcal{A}$ gives to $\mathcal{B}$ two messages, M_0 and M_1, and the challenge identity I^*. $\mathcal{B}$ flips a random coin $\sigma \xleftarrow{R} \{0,1\}$ and returns the ciphertext:

$$c_2^* \xleftarrow{R} \{0,1\}^\mu \qquad c_3^* = (g_{p_1}^z g_{p_2}^{x_2})^{xI^*+y}$$
$$c_4^* = g_{p_1}^z g_{p_2}^{x_2} \qquad c_5 = T_\nu$$
$$C = e(s_2^*, c_4^*) e(s_3^*, c_3^*)(c_5^*)^{s_1^*} \qquad c_1^* = M_\sigma \oplus \mathsf{ext}(C, c_2^*)$$

where (s_1^*, s_2^*, s_3^*) is the normal secret key for I^* created in Phase 1.

This sets $z_c = xI^* + y$. Since the values of x, y matter only modulo p_1 and z_c matters only modulo p_2, there is no correlation between them and z_c seems randomly chosen for any adversary.

If $T = e(g_{p_1}, g_{p_1})^{\beta z}$, then this is a semi-functional ciphertext of message M_σ. This is because the term $C = e(s_2^*, c_4^*) e(s_3^*, c_3^*)(c_5^*)^{s_1^*} = e(g,g)^{\alpha z}$. Therefore this game is the game KG_K. If $T = e(g_{p_1}, g_{p_1})^{\beta w}$, then $\mathcal{B}$ creates an invalid semi-functional ciphertext, because $C = e(g,g)^{\alpha z} \cdot e(g,g)^{\beta(w-z)t^*}$. This means that $\mathcal{A}$ played the game Final. Therefore $\mathcal{B}$ breaks Assumption 3 with advantage $\epsilon/(2L)$. □

LEMMA 5.9. *For any PPT adversary $\mathcal{A}$ it holds that*

$$\mathsf{Adv}^{\mathsf{Final}}_{\mathcal{A},\Sigma} \leq 2\epsilon_{\mathsf{ext}}$$

The proof is the same as lemma's 3.2 if we observe that $C = e(g,g)^{\alpha r} e(g,g)^{\beta(w-z)t^*}$ in game Final, i.e. a random group element of $\mathbb{G}_{T,1}$, the subgroup of $\mathbb{G}_T$ of order p_1.

THEOREM 5.10. *If Assumptions 1,2,3 hold and the extractor's second parameter ϵ_{ext} used in Σ is negligible in n, then our system is l-leakage secure, where $l = \log|\mathbb{G}_{T,1}| - k$ and k is the extractor's first parameter.*

PROOF. Let's say that $\epsilon_1(n), \epsilon_2(n), \epsilon_3(n)$ are the maximum advantages over all attackers on assumptions 1,2,3, respectively. Then for any attacker $\mathcal{A}$ on our system we have the following:

Lemma	Result	Range
5.3	$\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Restricted}}_{\mathcal{A},\Sigma} \leq 4\max(\epsilon_1, \epsilon_2)$	
5.4	$\mathsf{Adv}^{\mathsf{Restricted}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Leak}_0}_{\mathcal{A},\Sigma} \leq 2\epsilon_1$	
5.5	$\mathsf{Adv}^{\mathsf{Leak}_{i-1}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Leak}_i}_{\mathcal{A},\Sigma} \leq 2L\epsilon_2$	$1 \leq i \leq L-1$
5.6	$\mathsf{Adv}^{\mathsf{Leak}_{L-1}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{KG}_1}_{\mathcal{A},\Sigma} \leq 2L\epsilon_2$	
5.7	$\mathsf{Adv}^{\mathsf{KG}_{i-1}}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{KG}_i}_{\mathcal{A},\Sigma} \leq 2L\epsilon_2$	$2 \leq i \leq K$
5.8	$\mathsf{Adv}^{\mathsf{KG}_K}_{\mathcal{A},\Sigma} - \mathsf{Adv}^{\mathsf{Final}}_{\mathcal{A},\Sigma} \leq 2L\epsilon_3$	
5.9	$\mathsf{Adv}^{\mathsf{Final}}_{\mathcal{A},\Sigma} \leq 2\epsilon_{\mathsf{ext}}$	

By adding all the inequalities we get that:

$$\mathsf{Adv}^{\mathsf{CpaLeak}}_{\mathcal{A},\Sigma} \leq 4\max(\epsilon_1, \epsilon_2) + 2\epsilon_1 + 2L(L+K-1)\epsilon_2 + 2L\epsilon_3 + 2\epsilon_{\mathsf{ext}}$$

If the premise of the theorem is true, then all ϵ_1, ϵ_2, ϵ_3, ϵ_{ext} are negligible functions of n. Since $L(n), K(n)$ are polynomials of n, we conclude that the advantage of all PPT attackers for our system is negligible. □

Leakage Summary Here the amount of leakage is $l \leq \log|\mathbb{G}_{T,1}| - \omega(\log n) - m$. Thus if we assume that elements

in $\mathbb{Z}_N$ and in the composite order group $\mathbb{G}_T$ are represented with $3n$ bits, we get that the fraction of the secret key that can be leaked is slightly less than 1/9.

6. FUTURE DIRECTIONS

Improve Leakage Fraction A promising direction is to improve the leakage allowed from each secret key as the fraction of its size. It seems that our results can be generalized by using multiple tags in the secret key, but the security analysis is more complicated.

Multiple-key Leakage In all leakage-resilient IBE systems, including the ones presented in this paper, leakage is allowed from only one secret key per identity. Although, this can be easily achieved by generating the randomness of the key generation algorithm using a pseudo-random generator, leakage from multiple keys might be useful in HIBE and ABE systems, based on IBE constructions. In these cases different secret keys have to be generated for the same identities, and as a result it is more difficult to apply leakage-resilient techniques.

Master Secret Key Leakage As we saw no leakage is allowed from the master secret key of our systems. We assumed that it is totally hidden from the adversary. It is an interesting open question if there exist IBE systems resilient to MSK leakage. Since there is a generic transformation of any IBE system to a signature scheme having as signing key the master secret key of the IBE system, MSK-leakage-resilient systems will provide constructions of leakage-resilient signature schemes.

HIBE Finally it is an interesting open question whether leakage-resilient HIBE systems exist. Or how existing techniques (such as tagging) can be applied to this setting.

7. REFERENCES

[1] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. *EUROCRYPT*, 2010.

[2] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[3] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *STOC*, pages 30–41, 1977.

[4] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[5] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[6] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–341, 2005.

[7] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[8] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.

[9] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

[10] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[11] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.

[12] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.

[13] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[14] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377, 1982.

[15] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.

[16] S. Hohenberg and Brent Waters. Constructing verifiable random functions with large input spaces. *EUROCRYPT*, 2010.

[17] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[18] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.

[19] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

[20] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, pages 455–479, 2010.

[21] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.

[22] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

[23] Noam Nisan. Extracting randomness: How and why a survey. In *IEEE Conference on Computational Complexity*, pages 44–58, 1996.

[24] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[25] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.