Achieving Leakage Resilience Through Dual System Encryption

Allison Lewko[⋆], Yannis Rouselakis, and Brent Waters[⋆⋆]

The University of Texas at Austin
{alewko,jrous,bwaters}@cs.utexas.edu

Abstract. In this work, we show that strong leakage resilience for cryptosystems with advanced functionalities can be obtained quite naturally within the methodology of dual system encryption, recently introduced by Waters. We demonstrate this concretely by providing fully secure IBE, HIBE, and ABE systems which are resilient to bounded leakage from each of many secret keys per user, as well as many master keys. This can be realized as resilience against continual leakage if we assume keys are periodically updated and no (or logarithmic) leakage is allowed during the update process. Our systems are obtained by applying a simple modification to previous dual system encryption constructions: essentially this provides a generic tool for making dual system encryption schemes leakage-resilient.

1 Introduction

Defining and achieving the right security models is crucial to the value of provably secure cryptography. When security definitions fail to encompass *all* of the power of potential attackers, systems which are proven "secure" may actually be vulnerable in practice. It is often not realistic or desirable to address such problems solely at the implementation level. Instead, the ultimate goal of cryptography should be to provide efficient systems which are proven secure against the largest possible class of potential attackers. Additionally, these systems should provide the most advanced functionalities available.

Recently, much progress has been made in obtaining increasingly complex systems with stronger security guarantees. The emergence of *leakage-resilient cryptography* has led to constructions of many cryptographic primitives which can be proven secure even against adversaries who can obtain limited additional information about secret keys and other internal state. This line of research is motivated by a variety of side-channel attacks [46, 13, 7, 12, 53, 8, 47, 58, 33, 41],

[⋆] Supported by a National Defense Science and Engineering Graduate Fellowship.

[⋆⋆] Supported by NSF CNS-0915361, and CNS-0952692, Air Force Office of Scientific Research (AFO SR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO), Department of Homeland Security Grant 2006-CS-001-000001-02 (subaward 641), a Google Faculty Research award, and the Alfred P. Sloan Foundation.

which allow attackers to learn partial information about secrets by observing physical properties of a cryptographic execution such as timing, power usage, etc. The cold-boot attack [41] allows an attacker to learn information about memory contents of a machine even after the machine is powered down.

Leakage-resilient cryptography models a large class of side-channel attacks by allowing the attacker to specify an efficiently computable leakage function f and learn the output of f applied to the secret key and possibly other internal state at specified moments in the security game. Clearly, limits must be placed on f to prevent the attacker from obtaining the entire secret key and hence easily winning the game. One approach is to bound the total number of bits leaked over the lifetime of the system to be significantly less than the bit-length of the secret key. Another approach is to continually refresh the secret key and bound the leakage between each update (this is called "continual leakage"). Both of these approaches have been employed successfully in a variety of settings, yielding constructions of stream ciphers, signatures, symmetric key encryption, public key encryption, and identity-based encryption (IBE) which are leakage-resilient under various models of leakage [52, 45, 31, 57, 26, 1, 2, 32, 29, 24, 19, 30, 3, 15, 21, 16, 25].

Concurrently, the methodology of dual system encryption has emerged as a useful tool for improving the security guarantees for efficient cryptosystems with advanced functionalities like identity-based encryption (IBE), hierarchical identity-based encryption (HIBE), attribute-based encryption (ABE) [63, 50, 48]. These works provide efficient systems with short parameters which are proven fully secure in the standard model under static assumptions. Previous constructions of IBE and HIBE either used random oracles, had large parameters, were only proven selectively secure (a weaker model of security where the attacker must declare its target immediately instead of choosing it adaptively in the course of the security game), or relied on "q-based" assumptions (where the size of the assumption depends on the number of the attacker's queries) [14, 22, 37, 18, 9, 10, 61, 11, 34, 36, 35]. All previous constructions of ABE were only proven selectively secure [59, 40, 20, 6, 55, 39, 62]. Like leakage resilience, moving from selectively secure systems to fully secure systems is important because it results in security against a more powerful class of attackers.

Our Contribution In this work, we show that the techniques of dual system encryption naturally lead to leakage resilience. We demonstrate this by providing leakage-resilient constructions of IBE, HIBE, and ABE systems which retain all of the desirable features of dual system constructions, like full security from static assumptions and close resemblance to previous selectively secure schemes. We present our combination of dual system encryption and leakage resilience as a convenient abstraction and reduce proving security to the establishment of three properties.

Our approach not only combines the benefits of dual system encryption and leakage resilience, but also qualitatively improves upon the leakage tolerance of previous leakage-resilient IBE schemes [16, 2, 21]. In particular, our IBE system can tolerate leakage on the master key, as well as leakage on several keys for each identity (this can be viewed as continual leakage, where secret keys are periodically updated and leakage is allowed only *between* updates, and not *during* updates).[1] The IBE schemes of [2, 21] only allow bounded leakage on one secret key per identity, and allow no leakage on the master key. The IBE scheme of [16] allows bounded leakage on each of many keys per identity, but allows no leakage on the master key.

We develop a simple and versatile methodology for modifying a dual system encryption construction and proof to incorporate strong leakage resilience guarantees. The change to the constructions is minimal, and can be viewed as the adjoining of a separate piece which does not interfere with the intuitive and efficient structure of the original system. Essentially, we show that dual system encryption and leakage resilience are highly compatible, and their combination results in the strongest security guarantees available for cryptosystems with advanced functionalities, with no sacrifice of efficiency.

Our Techniques In a dual system encryption scheme, keys and ciphertexts can each take on two forms: normal and semi-functional. Normal keys can decrypt both forms of ciphertexts, while semi-functional keys can only decrypt normal ciphertexts. In the real security game, the ciphertext and all keys are normal. Security is proven by a hybrid argument, where first the ciphertext is changed to semi-functional, and then the keys are changed to semi-functional one by one. We must prove that the attacker cannot detect these changes. Finally, we arrive at a game where the simulator need only produce semi-functional objects, which cannot correctly decrypt. This greatly reduces the burden on the simulator and allows us to now prove security directly.

There is an important challenge inherent in this technique: when we argue the indistinguishability of games where a certain key is changing from normal to semi-functional, it is crucial that the simulator cannot determine the nature of this key for itself by test decrypting a semi-functional ciphertext. However, the simulator should also be prepared to make a semi-functional ciphertext for *any* identity and to use *any* identity for this particular key. This challenge is overcome by allowing the simulator to make *nominal* semi-functional keys: these are keys that are distributed like ordinary semi-functional keys in the attacker's view, but in the simulator's view they are correlated with the challenge ciphertext, so that if the simulator tries to decrypt a semi-functional ciphertext, decryption will always succeed, and hence will not reveal whether the key is normal or nominally semi-functional.

To keep nominal semi-functionality hidden from the attacker's view, previous dual system encryption constructions relied crucially on the fact that the attacker cannot ask for a key capable of decrypting the challenge ciphertext. When we add leakage to this framework, the attacker is now able to ask for leakage on keys which are capable of decrypting the challenge ciphertext: hence we need a

[1] For simplicity, we present our system as allowing no leakage during key updates, but our system can tolerate leakage which is logarithmic in terms of the security parameter using the same methods employed in [16].

new mechanism to hide nominal semi-functionality from attackers who can leak on these keys.

We accomplish this by expanding the semi-functional space to form $n + 2$ dimensional vectors, where $n \geq 3$ is a parameter determining the leakage tolerance. Nominality now corresponds to the vector in the semi-functional space of the key being orthogonal to the vector in the semi-functional space of the ciphertext. Because the leakage function on the key must be determined *before* the challenge ciphertext is revealed, an attacker whose leakage is suitably bounded cannot distinguish orthogonal vectors from uniformly random vectors in this context (this is a corollary of the result from [16], which shows that "random subspaces are leakage-resilient"). Hence, the attacker cannot distinguish leakage on a nominally semi-functional key from leakage on an ordinary semi-functional key. This allows us to obtain leakage resilience within the dual system encryption framework.

Comparison to Previous Techniques One of the leakage-resilient IBE constructions of [21] also applied the dual system encryption methodology, but ultimately relied on the technique of hash proof systems [23, 52, 2] to obtain leakage resilience, instead of deriving leakage resilience from the dual system encryption methodology itself, as we do in this work. More precisely, they used the dual system encryption framework to allow the simulator to produce keys incapable of decrypting the challenge ciphertext, but did not apply dual system encryption to handle leakage on keys which are capable of decrypting the challenge ciphertext. Instead, they relied on a hash proof mechanism for this part of the proof. This leads them to impose the restriction that the attacker can only leak from one key for the challenge identity, and no leakage on the master key is allowed. Essentially, their application of dual system encryption is "orthogonal" to their techniques for achieving leakage resilience. In contrast, our techniques allow us to handle all key generation and leakage queries *within the dual system encryption framework*, eliminating the need for a separate technique to achieve leakage resilience. This enables us to allow leakage from multiple keys which can decrypt the challenge ciphertext, as well as leakage from the master key.

The leakage-resilient IBE construction of [16] in the continual leakage model relies on selective security to allow the simulator to produce the keys incapable of decrypting challenge ciphertext. This is accomplished with a partitioning technique. Their technique for handling leakage on secret keys for the challenge identity is more similar to ours: they produce these keys and ciphertext in such a way that each is independently well-distributed, but the keys for the challenge identity exhibit degenerate behavior relative to the challenge ciphertext. This correlation, however, is information-theoretically hidden from the adversary because the leakage per key is suitably bounded. We employ a similar information-theoretic argument to hide nominal semi-functionality of leaked keys from the attacker's view. However, their technique does not quite fit our dual system encryption framework, and only achieves selective security in their implementation, with no leakage allowed from the master key.

1.1 Related Work

Leakage resilience has been studied in many previous works, under a variety of leakage models [60, 56, 45, 3, 19, 24, 30, 28, 44, 29, 52, 1, 2, 17, 26, 31, 42, 51, 57, 32, 15, 27, 16, 25]. Exposure-resilient cryptography [17, 28, 44] addressed adversaries who could learn a subset of the bits representing the secret key or internal state. Subsequent works have considered more general leakage functions. Micali and Reyzin [51] introduced the assumption that "only computation leaks information." In other words, one assumes that leakage occurs every time the cryptographic device performs a computation, but that any parts of the memory not involved in the computation do not leak. Under this assumption, leakage-resilient stream ciphers and signatures have been constructed [31, 57, 32]. Additionally, [43, 38] have shown how to transform any cryptographic protocol into one that is secure with continual leakage, assuming that only computation leaks information and also relying on a simple, completely non-leaking hardware device.

Since attacks like the cold-boot attack [41] can reveal information about memory contents in the absence of computation, it is desirable to have leakage-resilient constructions that do not rely upon this assumption. Several works have accomplished this by bounding the total amount of leakage over the lifetime of the system, an approach introduced by [1]. This has resulted in constructions of pseudorandom functions, signature schemes, public key encryption, and identity-based encryption [26, 52, 3, 45, 2, 21] which are secure in the presence of suitably bounded leakage. For IBE schemes in particular, this means that an attacker can leak a bounded amount of information from only *one secret key per user*. This does not allow a user to update/re-randomize his secret key during the lifetime of the system.

Recently, two works have achieved continual leakage resilience *without* assuming that only computation leaks information [16, 25]. Dodis, Haralambiev, Lopez-Alt, and Wichs [25] construct one-way relations, signatures, identification schemes, and authenticated key agreement protocols which are secure against attackers who can obtain leakage *between* updates of the secret key. It is assumed the leakage between consecutive updates is bounded in terms of a fraction of the secret key size, and also that there is no leakage during the update process. Brakerski, Kalai, Katz, and Vaikuntanathan [16] construct signatures, public key encryption schemes, and (selectively secure) identity-based encryption schemes which are secure against attackers who can obtain leakage between updates of the secret key, and also a very limited amount of leakage during updates and during the initial setup phase. The leakage between updates is bounded in terms of a fraction of the secret key size, while the leakage during updates and setup is logarithmically small as a function of the security parameter.

The dual system encryption methodology was introduced by Waters in [63]. It has been leveraged to obtain constructions of fully secure IBE and HIBE from simple assumptions [63], fully secure HIBE with short ciphertexts [50], fully secure ABE and Inner Product Encryption (IPE) [48], and fully secure functional encryption combining ABE and IPE [54].

Independently, Alwen and Ibraimi [4] have proposed a leakage resilient system for a special case of Attribute-Based Encryption, where the ciphertext policy is expressed as a DNF. Their work pursues a different technical direction to ours, and provides an interesting application of hash proof systems to the ABE setting. Security is proven from a "q-type" assumption.

2 Preliminaries

Notation We denote by $s \xleftarrow{\$} S$ the fact that s is picked uniformly at random from a finite set S and by $x, y, z \xleftarrow{\$} S$ that all x, y, z are picked independently and uniformly at random from S. We say that a function is *of constant output size* if the number of bits output by it is independent of the input. By $|x|$, we denote the size/number of bits of term x. Also, the special symbol $\perp$ is meant to serve as a unique dummy value in all our systems. Finally, by PPT we denote a probabilistic polynomial-time algorithm.

Complexity Assumptions To prove the security of our system, we will use three assumptions in composite order groups, also used in [50, 48]. These are static assumptions, which hold in the generic group model if finding a nontrivial factor of the group order is hard. The proof of this can be found in [50]. The first two of our assumptions belong to the class of General Subgroup Decision Assumptions described in [5]. The specific statement of the assumptions can be found in the full version [49].

2.1 Security Definition

In this section we assume familiarity with the main functionalities of the algorithms of an IBE system. Due to lack of space in this version we included the detailed definition only in the full version [49].

The security of our system is based on a game, called MasterLeak. It is a modified version of the usual IbeCpa security game. In that game, the attacker can make a polynomial number of **Keygen** queries for identities other than the challenge identity. Each of these queries returns a secret key of the requested identity. The main idea of our security game is to allow these queries and in addition allow leakage on the master key and secret keys of the challenge identity. The only restriction we impose is that it can not get leakage of more than ℓ_{MK} bits per master key (remember we can have many master keys) and ℓ_{SK} bits per secret key, where $\ell_{\text{MK}}, \ell_{\text{SK}}$ are parameters of the game.

The game starts with a setup phase, where the challenger runs the setup algorithm and gives the attacker the public parameters. It also gives the attacker a handle (i.e. reference) to the master key. We now allow the attacker to make three kinds of queries, called **Create**, **Leak**, and **Reveal**. With a **Create** query, the attacker asks the challenger to create a key and store it. The attacker supplies a handle that refers to a master key to be used in the key generation algorithm. Each such query returns a unique handle-reference to the generated key, so that the attacker can refer to it later and either apply a leakage function to it and/or ask for the entire key. The original master key (the one created in the **Setup** algorithm) gets a handle of 0.

Using a handle, the attacker can make a leakage query **Leak** on any key of its choice. Since all queries are adaptive (the attacker has the ability to leak from each key a few bits at the time, instead of requiring the leakage to occur all at once) and the total amount of leakage allowed is bounded, the challenger has to keep track of all keys leaked via these queries and the number of leaked bits from each key so far. Thus, it creates a set $\mathcal{T}$ that holds tuples of handles, identities, keys, and the number of leaked bits. Each **Create** query adds a tuple to this set and each **Leak** query updates the number of bits leaked.

The **Reveal** queries allow the attacker to get access to an entire secret key. They get as input a handle to a key and the challenger returns this secret key to the attacker. The obvious restriction is that the attacker cannot get a master key, since it would trivially break the system. For the same reason, no key for the challenge identity should be revealed and thus the challenger has to have another set to keep track of the revealed identities. We will denote this set by $\mathcal{R}$. We also note that the **Reveal** queries model the attacker's ability to "change its mind" in the middle of the game on the challenge identity. Maybe the attacker, after getting leakage from a secret key, decides that it is better to get the entire key via a **Reveal** query. Thus we achieve the maximum level of adaptiveness.

We now define our game formally. The security game is parameterized by a security parameter λ and two leakage bounds $\ell_{\mathrm{MK}} = \ell_{\mathrm{MK}}(\lambda)$, $\ell_{\mathrm{SK}} = \ell_{\mathrm{SK}}(\lambda)$. The master keys', secret keys' and identities' spaces are denoted by $\mathcal{MK}$, $\mathcal{SK}$, and $\mathcal{I}$, respectively. We assume that the handles' space is $\mathcal{H} = \mathbb{N}$. The game $\mathsf{MasterLeak}$ consists of the following phases:

Setup: The challenger makes a call to **Setup**(1^λ) and gets a master key MK and the public parameters PP. It gives PP to the attacker. Also, it sets $\mathcal{R} = \emptyset$ and $\mathcal{T} = \{(0, \epsilon, \mathrm{MK}, 0)\}$. Remember that $\mathcal{R} \subseteq \mathcal{I}$ and $\mathcal{T} \subseteq \mathcal{H} \times \mathcal{I} \times (\mathcal{MK} \cup \mathcal{SK}) \times \mathbb{N}$ (handles - identities - keys - leaked bits). Thus initially the set $\mathcal{T}$ holds a record of the original master key (no identity for it and no leakage so far). Also a handle counter H is set to 0.

Phase 1: In this phase, the adversary can make the following queries to the challenger. All of them can be interleaved in any possible way and the input of a query can depend on the outputs of all previous queries (adaptive security).

- **Create**(h, X): h is a handle to a tuple of $\mathcal{T}$ that must refer to a master key and X can be either an identity I or the empty string ϵ.
  The challenger initially scans $\mathcal{T}$ to find the tuple with handle h. If the identity part of the tuple is not ϵ, which means that the tuple holds a secret key of some identity, or if the handle does not exist, it responds with $\perp$.
  Otherwise, the tuple is of the form $(h, \epsilon, \mathrm{MK}', L)$. Then the challenger makes a call to **Keygen**$(\mathrm{MK}', X) \to K$ and adds the tuple $(H+1, X, K, 0)$ to the set $\mathcal{T}$. K is either a secret key for identity I or another master key depending on X. If X is an identity it returns a secret key and if X is the

empty string ϵ it returns another master key. See the full version [49] for a detailed definition. After that, it updates the handle counter to $H \leftarrow H + 1$.

- **Leak**(h, f): In this query, the adversary requests leakage from a key that has handle $h \in \mathbb{N}$ with a polynomial-time computable function f of constant output size[2] acting on the set of keys.
  The challenger scans $\mathcal{T}$ to find the tuple with the specified handle. It is either of the form (h, I, SK, L) or $\big(h, \epsilon, \mathrm{MK}', L\big)$ [3].
  In the first case, it checks if $L + |f(\mathrm{SK})| \leq \ell_{\mathrm{SK}}$. If this is true, it responds with $f(\mathrm{SK})$ and updates the L in the tuple with $L + |f(\mathrm{SK})|$. If the check fails, it returns $\perp$ to the adversary.
  If the tuple holds a master key MK', it checks if $L + \big|f(\mathrm{MK}')\big| \leq \ell_{\mathrm{MK}}$. If this is true, it responds with $f(\mathrm{MK}')$ and updates the L with $L + \big|f(\mathrm{MK}')\big|$. If the check fails, it returns $\perp$ to the adversary.
- **Reveal**(h): Now the adversary requests the entire key with handle h. The challenger scans $\mathcal{T}$ to find the requested entry. If the handle refers to a master key tuple, then the challenger returns $\perp$. Otherwise, we denote the tuple by (h, I, SK, L). The challenger responds with SK and adds the identity I to the set $\mathcal{R}$.

Challenge: The adversary submits a challenge identity $I^* \notin \mathcal{R}$ and two messages M_0, M_1 of equal size. The challenger flips a uniform coin $c \xleftarrow{\$} \{0, 1\}$ and encrypts M_c under I^* with a call to **Encrypt**(M_c, I^*). It sends the resulting ciphertext CT^* to the adversary.

Phase 2: This is the same as **Phase 1** with the restriction that the only queries allowed are **Create** and **Reveal** queries that involve a (non-master) secret key with identity different than I^*. The reason for forbidding **Leak** queries on a master key and on I^* is that the adversary can encode the entire decryption algorithm of CT^* as a function on a secret key, and thus win the game trivially if we allow these queries. For the same reason, the challenger can not give an entire secret key of I^* to the adversary and hence no **Reveal** queries involving I^* are allowed too. **Leak** queries on keys of identities other than I^* are useless, since the adversary can get the entire secret keys.

Guess: The adversary outputs a bit $c' \in \{0, 1\}$. We say it succeeds if $c' = c$.

The security definition we will use is the following:

Definition 1. *An IBE encryption system Π is $(\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$-master-leakage secure if for all PPT adversaries $\mathcal{A}$ it is true that*

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{MasterLeak}}(\lambda, \ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}) \leq \mathsf{negl}(\lambda)$$

[2] We apply this restriction so that the adversary does not get any "extra" information about the input; only the output bits of the function. This restriction is also present in other works (e.g. in [16] they use circuits as leakage functions).

[3] It can be the case that MK' is the original master key.

where $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{MasterLeak}}(\lambda, \ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$ *is the advantage of* $\mathcal{A}$ *in game* $\mathsf{MasterLeak}$ *with security parameter* λ *and leakage parameters* $\ell_{\mathrm{MK}} = \ell_{\mathrm{MK}}(\lambda), \ell_{\mathrm{SK}} = \ell_{\mathrm{SK}}(\lambda)$ *and is formally defined as*

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{MasterLeak}}(\lambda, \ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}) = \left| \Pr[\mathcal{A} \textit{ succeeds}] - \frac{1}{2} \right|,$$

where the probability is over all random bits used by the challenger and the attacker.

3 Dual System IBE

We now define dual system IBE schemes as an abstraction and define three security properties which will ensure leakage resilience. We show that these properties imply that a dual system IBE scheme is $(\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$-master-leakage secure [4].

3.1 Definition

A dual system IBE scheme Π_D has the following algorithms:

Setup$(1^\lambda) \to (\mathrm{PP}, \mathrm{MK})$ The setup algorithm takes in the security parameter, λ, and outputs the public parameters, PP, and a normal master key, MK.

Keygen$(\mathrm{MK}', X) \to K$ The key generation algorithm takes in a normal master key, MK$'$, and either an identity, I, or the empty string ϵ. In the first case, it outputs a normal secret key, SK, for the identity I, and in the second case, it outputs another normal master key, MK$''$.

Encrypt$(\mathrm{PP}, M, I) \to \mathrm{CT}$ The encryption algorithm takes in the public parameters PP, a message M, and an identity I, and outputs a normal ciphertext, CT.

Decrypt$(\mathrm{CT}, \mathrm{SK}) \to M$ The decryption algorithm takes in a ciphertext CT encrypted to identity I, and a secret key SK for identity I. It outputs the message M, unless both the key and the ciphertext are semi-functional.

KeygenSF$(\mathrm{MK}', X) \to \widetilde{K}$ The semi-functional key generation algorithm works in a similar way to **Keygen** but outputs semi-functional keys. If $X = I$, an identity, it outputs a semi-functional secret key, $\widetilde{SK}$ for identity I. If $X = \epsilon$, the empty string, it outputs a semi-functional master key, $\widetilde{MK}$.

Notice that this algorithm takes in a *normal* master key; not a semi-functional one. Also, this algorithm *need not be polynomial time computable*, in contrast to **Setup**, **Keygen**, **Encrypt**, and **Decrypt**.

EncryptSF$(\mathrm{PP}, M, I) \to \widetilde{\mathrm{CT}}$ The semi-functional encryption algorithm takes in the public parameters PP, a message M, and an identity I, and outputs a semi-functional ciphertext, CT. This algorithm *need not be polynomial time computable.*

[4] We choose not to include nominal semi-functionality as part of our abstraction, since one can use dual system encryption without employing this concept. For example, nominal semi-functionality was not used in [63].

3.2 Security Properties for Leakage Resilience

We now define three security properties for a dual system IBE scheme. For this, we define two additional games which are modifications of the MasterLeak game.

The first game, called MasterLeakC, is exactly the same as the MasterLeak game except that in the **Challenge** phase, the challenger uses **EncryptSF** instead of **Encrypt** to create a semi-functional ciphertext, and returns this to the adversary.

In the second new game, called MasterLeakCK, the challenger again uses **EncryptSF** for the challenge phase. However, the set of tuples $\mathcal{T}$ has a different structure. Each tuple holds for each key (master or secret) a normal and a semi-functional version of it. In this game, all keys leaked or given to the attacker are semi-functional. As we have noted above, the semi-functional key generation algorithm takes as input a normal master key. Thus the challenger stores the normal versions, as well the semi-functional ones so that it can use the normal versions of master keys as input to **Keygen** calls. [5] More precisely, the challenger additionally stores a semi-functional master key in tuple 0 by calling $\textbf{KeygenSF}(\mathrm{MK}, \epsilon)$ after calling **Setup**. Thereafter, for all $\textbf{Create}(h, X)$ queries, the challenger makes an additional call to $\textbf{KeygenSF}(\mathrm{MK}', X)$, where MK' is the *normal* version of the master key stored in tuple h. **Leak** and **Reveal** queries act always on the semi-functional versions of each key.

Finally, notice that the same attackers that play game MasterLeak can play games MasterLeakC and MasterLeakCK without any change in their algorithms - queries etc. The simulator answers them in a different way.

Semi-functional Ciphertext Invariance: We say that a dual system IBE scheme Π_D has *($\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}$)- semi-functional ciphertext invariance* if for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the MasterLeak game is negligibly close to the advantage of $\mathcal{A}$ in the MasterLeakC game.

Semi-functional Key Invariance: We say that a dual system IBE scheme Π_D has *($\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}$)-semi-functional key invariance* if for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the MasterLeakC game is negligibly close to the advantage of $\mathcal{A}$ in the MasterLeakCK game.

Semi-functional Security: We say that a dual system IBE scheme Π_D has *($\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}$)-semi-functional security* if for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the MasterLeakCK game is negligible.

The proof of the following theorem is straightforward, and can be found in the full version [49].

Theorem 1. *If a dual system IBE scheme Π_D =(**Setup**, **Keygen**, **Encrypt**, **Decrypt**, **KeygenSF**, **EncryptSF**) has ($\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}$)-semi-functional ciphertext invariance, ($\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}$)-semi-functional key invariance, and ($\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}$)-semi-functional security, then Π =(**Setup**, **Keygen**, **Encrypt**, **Decrypt**) is a ($\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}}$)-master-leakage secure IBE scheme.*

[5] As one should notice, we will never use the normal versions of non-master keys in this game. However, we have them here because we will need them in the game of the next section and when we move to the HIBE setting.

3.3 An Alternate Security Property

We additionally define a property called *One Semi-functional Key Invariance*. In the full version we will show that this implies semi-functional key invariance, and so can be substituted for semi-functional key invariance in proving that a system is $(\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$-master-leakage secure. The motivation for this is that proving semi-functional key invariance directly will often involve a hybrid argument, and defining one semi-functional key invariance allows us to include this hybrid as part of our abstraction and hence avoid repeating it for each system.

To define this property, we first define one more variation of our security game, called $\mathsf{MasterLeak}_b$. This is similar to the $\mathsf{MasterLeakCK}$ game, with the main difference being that the attacker can choose on which version of each key to leak or reveal. In other words, on the first leakage or reveal query on a key of the augmented set $\mathcal{T}$, the attacker tells the challenger whether it wants the normal or the semi-functional version of the key. In order for the challenger to keep track of the attacker's choice on each key, we further augment each tuple of $\mathcal{T}$ with a lock-value denoted by $V \in \mathbb{Z}$ that can take one of the three values $\{-1, 0, 1\}$:

- If $V = -1$ the attacker has not made a choice on this key yet and the key is "unlocked". This is the value the tuple gets, in a **Create** query.
- If $V = 0$ the attacker chose to use the normal version of the key on the first leakage or reveal query on it. All subsequent **Leak** and **Reveal** queries act on the normal version.
- If $V = 1$ the attacker chose the semi-functional version and the challenger works as above with the semi-functional version.

To summarize, each tuple is of the form $(h, X, K, \widetilde{K}, L, V)$ i.e. handle - identity or empty string - normal key - semi-functional key - leakage - lock. For example, the original master key is stored at the beginning of the game in the tuple $(0, \epsilon, \mathrm{MK}, \mathbf{KeygenSF}(\mathrm{MK}, \epsilon), 0, -1)$.

At some point, the attacker must decide on a *challenge key* which is "unlocked", $V = -1$, and tell this to the challenger. The challenger samples a uniformly random bit $b \xleftarrow{\$} \{0, 1\}$ and sets $V = b$. Therefore, the attacker has access to either the normal (if $b = 0$) or the semi-functional (if $b = 1$) version of this key via **Leak** and **Reveal** queries. We note that if the attacker did not make a choice for the original master key in tuple 0, it can choose this master key as the challenge key.

The attacker is then allowed to resume queries addressed to either normal or semi-functional keys, with the usual restrictions (i.e. no leakage or reveal queries on keys capable of decrypting the challenge ciphertext after the attacker has seen the challenge ciphertext).

One Semi-functional Key Invariance: We say that a dual system IBE scheme Π_D has $(\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$-*one semi-functional key invariance* if, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the $\mathsf{MasterLeak}_b$ game with $b = 0$ is negligibly close to the advantage of $\mathcal{A}$ in the $\mathsf{MasterLeak}_b$ game with $b = 1$.

The proof of the following theorem can be found in the full version [49].

Theorem 2. *If a dual system IBE scheme Π_D has $(\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$-one semi-functional key invariance, then it also has $(\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$-semi-functional key invariance.*

4 Master-Leakage Secure IBE Scheme

Our IBE scheme is an augmented version of the Lewko-Waters IBE [50], designed to sustain master and secret key leakage from an arbitrary number of keys. To hide nominal semi-functionality in the attacker's view, we add vectors of dimension n to the front of the ciphertexts and secret keys of the LW system. Notice in the construction below that the last two elements of our secret keys and ciphertexts are very similar to the elements in the LW system (which is essentially a composite order version of the selectively-secure Boneh-Boyen IBE system [9]). Nominal semi-functionality now corresponds to the vector of exponents of the semi-functional components of the key being orthogonal to the vector of exponents of the semi-functional components of the ciphertext. We can use the algebraic lemma of [16] to assert that this orthogonality is hidden from attackers with suitably bounded leakage. Finally, to allow leakage on the master key, we designed the master key to be similar in form to regular secret keys.

Like the original LW scheme, our system uses a bilinear group whose order is the product of three distinct primes (additional background on composite order bilinear groups can be found in the full version [49]). The role of the first prime order subgroup is to "carry" the necessary information of the plaintext message and the secret information of each user or the master authority. The second subgroup is used only in the proof to provide semi-functionality. The third subgroup is used to additionally randomize secret keys. Each of these components is orthogonal to the other two under the pairing operation.

In the construction below, we will use angle brackets to denote vectors and parentheses to denote collections of elements of different types. The dot product of vectors is denoted by $\cdot$ and component-wise multiplication is denoted by $*$. For a group element $u \in \mathbb{G}$ and a vector $\vec{a} \in \mathbb{Z}_N^n$, we define $u^{\vec{a}}$ to be $\langle u^{a_1}, u^{a_2}, \ldots, u^{a_n} \rangle$. We also define a pairing operation of vectors in $\mathbb{G}^n$: For $\vec{v}_1 = \langle v_{11}, v_{12}, \ldots, v_{1n} \rangle \in \mathbb{G}^n$ and $\vec{v}_2 = \langle v_{21}, v_{22}, \ldots, v_{2n} \rangle \in \mathbb{G}^n$, their pairing is $e_n(\vec{v}_1, \vec{v}_2) := \prod_{i=1}^n e(v_{1i}, v_{2i}) \in \mathbb{G}_T$, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is the bilinear mapping of $\mathbb{G}$ and the product is the group operation of $\mathbb{G}_T$.

4.1 Construction

Our dual system IBE scheme consists of the following algorithms:

Setup(1^λ) The setup algorithm generates a bilinear group $\mathbb{G}$ of composite order $N = p_1 p_2 p_3$, where p_1, p_2, p_3 are three different λ_1,λ_2,λ_3-bit prime numbers respectively[6]. The subgroup of order p_i in $\mathbb{G}$ is denoted by $\mathbb{G}_i$. We assume that the identities of users in our system are elements of $\mathbb{Z}_N$.

[6] The three λ's depend on the security parameter and are chosen appropriately to get a better leakage fraction (see Section 5 for details).

We let n be a positive integer greater than or equal to 2. The value of n can be varied - higher values of n will lead to a better fraction of leakage being tolerated (see Section 5), while lower values of n will yield a system with fewer group elements in the keys and ciphertexts.

The algorithm picks 3 random elements $\langle g_1, u_1, h_1 \rangle \in \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1$ and one random element $g_3 \in \mathbb{G}_3$. It also picks $n+1$ random exponents $\langle \alpha, x_1, x_2, \ldots, x_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$. It picks $\langle r, y_1, y_2, \ldots, y_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, a random vector $\vec{\rho} = \langle \rho_1, \ldots, \rho_{n+2} \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$, and a random element $\rho_{n+3} \overset{\$}{\leftarrow} \mathbb{Z}_N$. It outputs the following public parameters and master key:

$$\mathrm{PP} = (N, g_1, g_3, u_1, h_1, e(g_1, g_1)^{\alpha}, g_1^{x_1}, g_1^{x_2}, \ldots, g_1^{x_n})$$

$$\mathrm{MK} = \left(\vec{K^*}, K^*\right) = \left(\left\langle g_1^{y_1}, \ldots, g_1^{y_n}, g_1^{\alpha} h_1^{-r} \prod_{i=1}^{n} g_1^{-x_i y_i}, g_1^{r} \right\rangle * g_3^{\vec{\rho}}, u_1^{r} g_3^{\rho_{n+3}}\right)$$

Keygen(MK, PP, X) We first consider when $X = \epsilon$, the empty string. Then this algorithm re-randomizes the master key by picking another $\langle r', y_1', y_2', \ldots, y_n' \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, a random vector $\vec{\rho'} = \langle \rho_1', \ldots, \rho_{n+2}' \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$, and a random element $\rho_{n+3}' \overset{\$}{\leftarrow} \mathbb{Z}_N$. If $\mathrm{MK} = \left(\vec{K^*}, K^*\right)$, it outputs the new (same-sized) master key:

$$\mathrm{MK}' = \left(\vec{K'}, K'\right) = \left(\vec{K^*} * \left\langle g_1^{y_1'}, \ldots, g_1^{y_n'}, h_1^{-r'} \prod_{i=1}^{n} g_1^{-x_i y_i'}, g_1^{r'} \right\rangle * g_3^{\vec{\rho'}}, K^* u_1^{r'} g_3^{\rho_{n+3}'}\right)$$

If $X = I \in \mathbb{Z}_N$, an identity, the algorithm picks $n+1$ random exponents $\langle r', z_1, z_2, \ldots, z_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$. Also it picks $\vec{\rho'} \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$ and outputs the secret key:

$$\mathrm{SK} = \vec{K_1} = \vec{K^*} * \left\langle g_1^{z_1}, g_1^{z_2}, \ldots, g_1^{z_n}, (K^*)^{-I} (u_1^{I} h_1)^{-r'} \prod_{i=1}^{n} g_1^{-x_i z_i}, g_1^{r'} \right\rangle * g_3^{\vec{\rho'}}$$

The terms $g_1^{-x_i y_i'}$ and $g_1^{-x_i z_i}$ above are calculated by using the g^{x_i} terms of PP.

It is very important to notice that with knowledge of α alone, one can create properly distributed secret keys, because the random terms $r, y_1, \ldots, y_n, \rho_{n+3}, \vec{\rho}$ of the master key are all masked by the random terms $r', z_1, \ldots, z_n, \vec{\rho'}$ generated by the algorithm. However, instead of storing α, the master authority now stores $n+3$ elements of $\mathbb{G}$.

Encrypt(M, I) The encryption algorithm picks $s \overset{\$}{\leftarrow} \mathbb{Z}_N$ and outputs the ciphertext:

$$\begin{aligned} \mathrm{CT} &= \left(C_0, \vec{C_1}\right) = \\ &= \left(M \cdot (e(g_1, g_1)^{\alpha})^{s}, \left\langle (g_1^{x_1})^{s}, \ldots, (g_1^{x_n})^{s}, g_1^{s}, (u_1^{I} h_1)^{s} \right\rangle\right) \in \mathbb{G}_T \times \mathbb{G}^{n+2} \end{aligned}$$

Decrypt(CT, SK) To calculate the blinding factor $e(g_1, g_1)^{\alpha s}$, one computes $e_{n+2}(\vec{K_1}, \vec{C_1})$ and the message is computed as $M = \frac{C_0}{e_{n+2}(\vec{K_1}, \vec{C_1})}$.

4.2 Semi-Functionality

All the ciphertexts, master keys, and secret keys generated by the above algorithms are *normal*, where by normal we mean that they have no $\mathbb{G}_2$ parts. On the other hand, a *semi-functional* key or ciphertext has $\mathbb{G}_2$ parts. We let g_2 denote a generator of $\mathbb{G}_2$. The remaining algorithms of our dual system IBE are the following:

KeygenSF$(\mathrm{MK}, X) \to \widetilde{K}$ This algorithm calls first the normal key generation algorithm **Keygen**(MK, X) to get a normal key $\mathrm{MK} = \left(\vec{K^*}, K^*\right)$ or $\mathrm{SK} = \vec{K_1}$, depending on X.

In the former case, it picks $\vec{\theta} \xleftarrow{\$} \mathbb{Z}_N^{n+2}$ and $\theta \xleftarrow{\$} \mathbb{Z}_N$ and outputs

$$\widetilde{\mathrm{MK}} = \left(\vec{K^*} * g_2^{\vec{\theta}}, K^* g_2^{\theta}\right).$$

In the latter case, it picks $\vec{\gamma} \xleftarrow{\$} \mathbb{Z}_N^{n+2}$ and outputs

$$\widetilde{\mathrm{SK}} = \vec{K_1} * g_2^{\vec{\gamma}}.$$

EncryptSF$(M, I) \to \widetilde{\mathrm{CT}}$ This algorithm calls first the normal encryption algorithm **Encrypt**(M, I) to get the ciphertext $\mathrm{CT} = \left(C_0, \vec{C_1}\right)$. Then it picks $\vec{\delta} \xleftarrow{\$} \mathbb{Z}_N^{n+2}$ and outputs

$$\widetilde{\mathrm{CT}} = \left(C_0, \vec{C_1} * g_2^{\vec{\delta}}\right).$$

We call the three terms $\left(\vec{\theta}, \theta\right), \vec{\gamma}, \vec{\delta}$ the *semi-functional parameters* of the master key, secret key, and ciphertext, respectively. The semi-functional keys are partitioned in *nominal* semi-functional keys and in *truly* semi-functional keys, with respect to a specific semi-functional ciphertext. In short, a nominal secret key can correctly decrypt the ciphertext (by using **Decrypt**), while a nominal master key can generate a semi-functional secret key that correctly decrypts the ciphertext.

As a result, a semi-functional secret key of identity I_k with parameters $\vec{\gamma}$ is nominal with respect to a ciphertext for identity I_c with parameters $\vec{\delta}$ if and only if $\vec{\gamma} \cdot \vec{\delta} = 0 \bmod p_2$ and $I_k = I_c$.

It is easy to see that only then the decryption is correct, because we get an extra term $e(g_2, g_2)^{\vec{\gamma} \cdot \vec{\delta}}$ by the pairing. A semi-functional master key with parameters $\vec{\theta}, \theta$ is nominal with respect to a ciphertext for identity I with parameters $\vec{\delta}$ if and only if $\vec{\delta} \cdot \left(\vec{\theta} + \langle 0, \ldots, 0, -I\theta, 0\rangle\right) = 0 \bmod p_2$.

The proof of the following theorem can be found in the full version [49].

Theorem 3. *Under our complexity assumptions and for* $(\ell_{\mathrm{MK}} = (n-1-2c)\log(p_2), \ell_{\mathrm{SK}} = (n-1-2c)\log(p_2))$*, where* $c > 0$ *is a fixed positive constant, our dual system IBE scheme is* $(\ell_{\mathrm{MK}}, \ell_{\mathrm{SK}})$*-master-leakage secure.*

Our HIBE and ABE constructions as well as their security proofs can also be found in the full version [49].

Scheme	Master Key	Secret Key
IBE	$\frac{n-1-2c}{n+3} \cdot \frac{1}{1+c_1+c_3}$	$\frac{n-1-2c}{n+2} \cdot \frac{1}{1+c_1+c_3}$
HIBE	$\frac{n-1-2c}{n+2+D-i} \cdot \frac{1}{1+c_1+c_3}$ for key of depth i in the hierarchy	
ABE	$\frac{n-1-2c}{n+2+\|U\|} \cdot \frac{1}{1+c_1+c_3}$	$\frac{n-1-2c}{n+2+\|S\|} \cdot \frac{1}{1+c_1+c_3}$

Table 1. c, c_1, c_3 are arbitrary positive constants and n is an integer greater than 2. For the HIBE scheme, D is the maximum depth of the hierarchy and i is the depth of the key in question. The master key is considered to be the root of the hierarchy tree and had depth 0. For the ABE scheme, $|U|$ is the total number of attributes in the system, and $|S|$ is the number of attributes of the key in question. Notice that in the ABE scheme we ignored the size of the representations of U and S. They are included in the keys, but they are considered public; thus not included in the leakage fraction.

5 Our Leakage Bound

Our systems allow the same absolute amount of leakage for both the master and the secret keys. That is, $\ell_{\text{MK}} = \ell_{\text{SK}} = (n-1-2c)\log p_2$ bits, where n is an arbitrary integer greater than or equal to 2 and c is a fixed positive constant. Notice that the leakage depends only on the size of the $\mathbb{G}_2$ subgroup, and not on the size of p_1 or p_3. Thus by varying the relative sizes of the $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ subgroups, we can achieve variable key sizes and allow different fractions of the key size to be leaked. We use the term "leakage fraction" to mean the number of bits allowed to be leaked from a key divided by the number of bits required to represent that key.

Recall that p_1, p_2, p_3 are primes of $\lambda_1, \lambda_2, \lambda_3$ bits, respectively, and $N = p_1p_2p_3$ is the order of our group $\mathbb{G}$. We assume that each group element is represented by approximately $\lambda_1 + \lambda_2 + \lambda_3 = \Theta(\log N)$ bits. Then, by fixing $\lambda_1 = c_1\lambda$, $\lambda_2 = \lambda$, and $\lambda_3 = c_3\lambda$, where λ is the security parameter and c_1, c_3 are arbitrary positive constants, we get that the leakage fractions of our systems are the values presented in the table above.

One notable property of our HIBE scheme is that the higher our keys are in the hierarchy, the less leakage is allowed from them. The master key which is at the top allows for a leakage fraction of $(n-1-2c)/((n+2+D)(1+c_1+c_3))$. This is because the base system we adapted, a HIBE system with short ciphertexts, has keys which contain more group elements for users which are higher in the hierarchy. This feature could be avoided by choosing a different base system.

The leakage fraction can be made arbitrarily close to 1 by modifying n, c_1 and c_3 (if we assume a fixed maximum depth for HIBE and a fixed universe size for ABE). Higher values of n give a better leakage fraction, but larger public parameters, keys, and ciphertexts. Smaller values of c_1, c_3 give a better leakage fraction, but also give fewer bits of security in the $\mathbb{G}_1$ and $\mathbb{G}_3$ subspaces as a function of λ. We must choose λ so that $c_1\lambda$ and $c_3\lambda$ are sufficiently large.

Acknowledgments

We are thankful to Joël Alwen and Yevgeniy Vahlis for useful observations.

References

1. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.
2. J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.
3. J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.
4. J. Alwen and L. Ibraimi. Leakage resilient ciphertext-policy attribute-based encryption. manuscript, 2010.
5. M. Bellare, B. Waters, and S. Yilek. Identity-based encryption secure under selective opening attack. In *TCC*, 2011.
6. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.
7. E. Biham, Y. Carmeli, and A. Shamir. Bug attacks. In *CRYPTO*, pages 221–240, 2008.
8. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, pages 513–525, 1997.
9. D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.
10. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.
11. D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, pages 440–456, 2005.
12. D. Boneh and D. Brumley. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.
13. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT*, pages 37–51, 1997.
14. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.
15. Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010.
16. Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.
17. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.
18. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.
19. D. Cash, Y. Z. Ding, Y. Dodis, W. Lee, R. J. Lipton, and S. Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In *TCC*, pages 479–498, 2007.
20. L. Cheung and C. Newport. Provably secure ciphertext policy abe. In *ACM Conference on Computer and Communications Security*, pages 456–465, 2007.

21. S. Chow, Y. Dodis, Y. Rouselakis, and B. Waters. Practical leakage-resilient identity-based encryption from simple assumptions. In *ACM Conference on Computer and Communications Security*, pages 152–161, 2010.
22. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.
23. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.
24. D. Di Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.
25. Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.
26. Y. Dodis, Y. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.
27. Y. Dodis and K. Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In *CRYPTO*, pages 21–40, 2010.
28. Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. In *EUROCRYPT*, pages 301–324, 2001.
29. S. Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.
30. S. Dziembowski and K. Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.
31. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.
32. S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.
33. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *CHES*, pages 251–261, 2001.
34. C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.
35. C. Gentry and S. Halevi. Hierarchical identity based encryption with polynomially many levels. In *TCC*, pages 437–456, 2009.
36. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM Symposium on Theory of Computing*, pages 197–206. ACM, 2008.
37. C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.
38. S. Goldwasser and G. Rothblum. Securing computation against continuous leakage. In *CRYPTO*, pages 59–79, 2010.
39. V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP (2)*, pages 579–591, 2008.
40. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.
41. A. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino, A. Feldman, J. Applebaum, and E. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.
42. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
43. A. Juma and Y. Vahlis. On protecting cryptographic keys against side-channel attacks. In *CRYPTO*, pages 41–58, 2010.

44. J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *FOCS*, pages 92–101, 2003.
45. J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.
46. P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.
47. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
48. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.
49. A. Lewko, Y. Rouselakis, and B. Waters. Achieving leakage resilience through dual system encryption. Cryptology ePrint Archive, Report 2010/438, 2010.
50. A. Lewko and B. Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, pages 455–479, 2010.
51. S. Micali and L. Reyzin. Physically observable cryptography. In *TCC*, pages 278–296, 2004.
52. M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
53. P. Q. Nguyen and I. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15(3):151–176, 2002.
54. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.
55. R. Ostrovksy, A. Sahai, and B. Waters. Attribute based encryption with non-monotonic access structures. In *ACM conference on Computer and Communications Security*, pages 195–203, 2007.
56. C. Petit, F.X. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *ASIACCS*, pages 56–65, 2008.
57. K. Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.
58. J. Quisquater and D. Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
59. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
60. F.X. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, pages 443–461, 2009.
61. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.
62. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008.
63. B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.