A Leakage-Resilient Spatial Encryption Scheme

Jill-Jênn Vie, Michel Abdalla, Crypto Team

August 23, 2011

This report is written in English because it is part of an upcoming research article.

The general context

A scheme is said leakage resilient if it remains secure even when an adversary is able to learn partial information about some secret values used throughout the lifetime of the system.

This recent area appeared because of the need to develop schemes that resist to side-channel attacks, such as power-consumption, fault or time analyses.

Today, there exist a few encryption schemes that implement this security model, notably a leakage-resilient identity-based encryption scheme (IBE): a public-key encryption scheme where the public keys are the identities, and a master key allows the generation of a secret key for each identity.

The research problem

There exists a hierarchical variant of the IBE scheme (HIBE), in which the identities form the nodes of a tree called hierarchy. Any identity can delegate secret keys to its successors and the root identity has a master key. Each node is associated with an identity vector, which represents the sequence of identities from the root to this node.

A spatial encryption scheme is a generalization of HIBE, in which keys are associated to affine spaces. In spatial encryption, any user possessing a secret key for an affine space can delegate secret keys for any of its affine subspaces, for the natural inclusion. To date, none of the existing constructions of spatial encryption are leakage resilient and the goal of this project is to propose one such construction.

Contributions

In this work, we propose the first leakage-resilient spatial encryption scheme. Our scheme works over bilinear groups of composite order, has constant-size ciphertexts, and is reasonably efficient. To achieve this goal, we adapted an existing leakage-resilient HIBE

construction due to Lewko, Rouselakis, and Waters in TCC 2011 to the spatial encryption setting. More precisely, we showed how to combine the Lewko-Rouselakis-Waters technique for achieving leakage resilience with a previous construction of spatial encryption due to Boneh and Hamburg in ASIACRYPT 2008.

Arguments supporting its validity

In order to show evidence that our new construction of leakage-resilient spatial encryption is sound, we also prove it secure in a formal security model. The proof of security is based on existing standard complexity assumptions over bilinear groups of composite order. As we show in the report, the resulting scheme remains secure even when the adversary learns partial information about the secret keys associated with each affine space, including the entire space.

Summary and future work

In this work, we have designed the first leakage-resilient spatial encryption scheme. Since spatial encryption includes identity-based and hierarchical identity-based encryption as special cases, previous solutions such as the Lewko-Rouselakis-Waters leakage-resilient HIBE construction can be seen as a particular instantiation of our construction. In addition to having constant-size ciphertexts and being reasonably efficient, our scheme is proven secure based on existing standard complexity assumptions over bilinear groups of composite order.

Unfortunately, as mentioned in the report, our scheme does not tolerate leakage during key updates. To address this problem and obtain a scheme with stronger security guarantees, a promising future research direction would be to adapt a recent result by Lewko, Lewko and Waters in STOC 2011 to the spatial encryption scenario.

A Leakage-Resilient Spatial Encryption Scheme

Jill-Jênn Vie
Supervisor: Michel Abdalla
Crypto Team

August 23, 2011

Contents

1 Introduction

When a theoretical scientist designs a cryptosystem, he is often unpleasantly surprised by the weaknesses its physical implementation can induce[1]. Indeed, history has proven that these attempts to break the system, called *side-channel attacks*, can be devastating.

In order to design schemes that can provably resist these attacks, a stronger notion of security has been established. A *leakage-resilient* scheme considers that an attacker is able to learn partial information about some secret values used throughout the lifetime of the system.

1.1 Leakage Models

Our achievement of leakage resilience will be based on the model of *memory attacks*, introduced by Akavia, Goldwasser, and Vaikuntanathan [AGV09]. According to it, the adversary can learn *arbitrary* information about the secret state of a system, by selecting polynomial-time computable functions $f_i : \{0,1\}^* \to \{0,1\}^{\lambda_i}$ and learning the value of f_i applied to the internal state of the system.

As described in [BSW11], there are two main variants of the memory attacks model, known as the bounded-leakage and the continual-leakage models.

In a *bounded-leakage* model, we define a *leakage bound* λ such that the overall amount of information learned by the adversary throughout the entire lifetime of the system is $\sum_i \lambda_i \leq \lambda$.

If a system is used continually for a sufficiently long time, then any static piece of information that stays unmodified on the system can eventually be recovered by the adversary. Thus, the secret keys of such systems must be periodically *refreshed*. Recently, Dodis et al. [DHLAW10] suggested the *continual-leakage* model, in which a scheme periodically self-refreshes its internal secret key, while the corresponding public key remains fixed. Therefore, it is the amount of leakage observed by the adversary in between any two successive refreshes which is bounded by λ. We can think of λ as a *leakage rate*.

1.2 Spatial Encryption

The goal of this work is to consider leakage resilience in the context of spatial encryption, which is a generic public-key cryptosystem first defined in [BH08] where vectors play the role of public keys and secret keys are associated to affine spaces. In a spatial encryption scheme, any user possessing a secret key for a certain space W_1 can *delegate* a secret key for a space W_2 included in W_1. A message encrypted for a certain vector $\boldsymbol{u}$ can be decrypted by any user possessing a secret key associated to a space that contains $\boldsymbol{u}$.

[1] Such as those due to a constant random number (cf. Sony's ECDSA code in PS3), but this is another problem.

In addition to proposing the concept of spatial encryption, Boneh and Hamburg [BH08] also provided a construction based on bilinear maps which is provably secure in the selective security model based on the BDDHE assumption ([BGW05]). In the selective secure model, the adversary decides on the vector upon which he wants to be challenged before seeing the public key of the scheme. Their work was later improved by Moriyama and Doi in [MD11] who proposed a fully secure (as opposed to selective) spatial encryption scheme in the standard model based on bilinear groups of composite order. In fact, the scheme that we propose in this work can be seen as a generalization of Moriyama-Doi spatial encryption scheme to leakage-resilient setting.

1.3 Outline

In Section 2, we will state our complexity assumptions. In Section 3, we will recall the formal definition of spatial encryption along with those of identity-based and hierarchical identity-based encryption, which are special cases of the former. In Section 4, we will present our leakage-resilient spatial encryption scheme. It is worth noting that, to date, none of the existing constructions of spatial encryption are leakage resilient and that our scheme is the first one to achieve this level of security. We will end this report with a few concluding remarks.

2 Preliminaries

2.1 Composite Order Bilinear Groups

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

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

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

2.2 Complexity Assumptions

To prove the security of our system, we will use the following assumptions in composite order groups, also used in [LOS+10, LW10, LRW11]. The first two of them belong to the class of Generalized Subgroup Decision Assumptions described in [BWY11]: in a bilinear group of order $N = p_1 \dots p_n$, for each subset $S \subseteq \{1, \dots, n\}$, there exists $G_S = G_{\prod_{i \in S} p_i}$ subgroup of order $\prod_{i \in S} p_i$. Let S_0, S_1 be two subsets. It is *hard to distinguish* a random element from G_{S_0} from a random element of G_{S_1}, even if one is given random elements from a family of (G_{Z_i}) such that each Z_i satisfy either $S_0 \cap Z_i = \varnothing = S_1 \cap Z_i$ or $S_0 \cap Z_i \neq \varnothing \neq S_1 \cap Z_i$.

Assumption 1. Given $\mathcal{D}_1 = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, g_3)$, no PPT (probabilistic polynomial-time) adversary has a non-negligible advantage in distinguishing

$$T_1 = g_1^a \text{ from } T_2 = g_1^a g_2^b$$

where $a, b \xleftarrow{R} \mathbb{Z}_N$.

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 is defined as:

$$\mathbf{Adv}_1^{\mathcal{A}}(\lambda) = \left|\Pr[\mathcal{A}(\mathcal{D}_1, T_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}_1, T_2) = 1]\right|$$

We say that Assumption 1 holds if for all PPT $\mathcal{A}$, $\mathbf{Adv}_1^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 2. Given $\mathcal{D}_2 = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, g_3, g_1^x g_2^y, g_2^{y'} g_3^{z'})$ where $x, y, y', z' \xleftarrow{R} \mathbb{Z}_N$, no PPT adversary has a non-negligible advantage in distinguishing

$$T_1 = g_1^a g_3^c \text{ from } T_2 = g_1^a g_2^b g_3^c$$

where $a, b, c \xleftarrow{R} \mathbb{Z}_N$.

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 is defined as:

$$\mathbf{Adv}_2^{\mathcal{A}}(\lambda) = \left|\Pr[\mathcal{A}(\mathcal{D}_2, T_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}_2, T_2) = 1]\right|$$

We say that Assumption 2 holds if for all PPT $\mathcal{A}$, $\mathbf{Adv}_2^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 3. Given $\mathcal{D}_3 = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, g_2, g_3, g_1^x g_2^y, g_1^{x'} g_2^{y'})$ where $x, y, x', y' \xleftarrow{R} \mathbb{Z}_N$, no PPT adversary has a non-negligible advantage in distinguishing

$$T_1 = \hat{e}(g_1, g_1)^{xx'} \in \mathbb{G}_T \text{ from } T_2 \xleftarrow{R} \mathbb{G}_T.$$

The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 is defined as:

$$\mathbf{Adv}_1^{\mathcal{A}}(\lambda) = \left|\Pr[\mathcal{A}(\mathcal{D}_3, T_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}_3, T_2) = 1]\right|$$

We say that Assumption 3 holds if for all PPT $\mathcal{A}$, $\mathbf{Adv}_3^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

3 Generalized Identity-Based Encryption

3.1 Identity-Based Encryption

An identity-based encryption system is a public-key cryptosystem which allows users to encrypt messages knowing only the recipient's identity and some public parameters. It consists of four PPT algorithms.

Setup$(1^\lambda) \to (\mathrm{PP}, \mathrm{MK})$ The setup algorithm takes an integer security parameter λ as input and outputs the public parameters PP and the original master key MK. In the rest of this report, all algorithms will take implicitly the security parameter and the public parameters as inputs.

KeyGen$(\mathrm{MK}', X) \to K$ The key generation algorithm takes in a master key MK' and either $X = I$, and identity, or $X = \varepsilon$, the empty string. In the former case, it outputs a secret key $K = \mathrm{SK}$, for the identity I. In the latter case, it outputs another master key, $K = \mathrm{MK}''$, such that $|\mathrm{MK}'| = |\mathrm{MK}''|$. This new master key can now be used instead of the original key in calls of **KeyGen**.

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

Decrypt$(\mathrm{CT}, \mathrm{SK}) \to M$ The decryption algorithm takes in a ciphertext CT for message M and a secret key SK. It outputs the message M.

The correctness requirement is that if the identity I used during the encryption of M is the same as the identity of the secret key used during decryption, then the output of **Decrypt** is the message M. That is, for all PP generated by a call to **Setup**, for all master keys MK' generated by applying the **KeyGen** algorithm and for all M, I,

$$\textbf{Decrypt}(\textbf{Encrypt}(M, I), \textbf{KeyGen}(\mathrm{MK}', I)) = M.$$

3.2 Hierarchical Identity-Based Encryption

In an HIBE system, identities form a structured hierarchy: a user can delegate keys to its subordinate identities, and thus decrypt any message encrypted to them.

We handle identity vectors $\boldsymbol{I} = (I_1, \ldots, I_j)$, and define the children of $\boldsymbol{I}$ as all vectors $\boldsymbol{I}||I_x$, where $||$ denotes concatenation and I_x is an identity. A user that has a secret key for vector $\boldsymbol{I}$ can create secret keys for all users that have identity vectors prefixed by $\boldsymbol{I}$. An example of hierarchy can be seen in Figure 1.

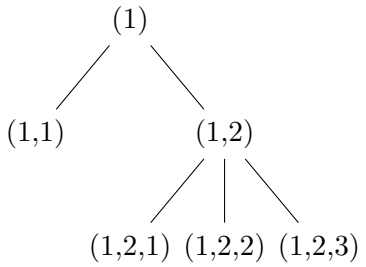

Figure 1: Vector identities in a hierarchy.

The depth D of the tree is the maximum length of the identity vectors and is called the depth of the hierarchy. The master key is formulated as a secret key for level 0, thus the corresponding identity vector is the empty vector.

$\left.\begin{array}{l}\textbf{Setup}(1^\lambda) \to (\mathrm{PP}, \mathrm{MK}) \\ \textbf{KeyGen}(\mathrm{MK}', \boldsymbol{I}) \to K \\ \textbf{Encrypt}(M, \boldsymbol{I}) \to \mathrm{CT}\end{array}\right\}$ have the same description than in the IBE scheme.

$\textbf{Delegate}(\boldsymbol{I}, \mathrm{SK}_{\boldsymbol{I}}, I_x) \to \mathrm{SK}_{\boldsymbol{I}||I_x}$ The delegation algorithm allows a user that has a secret key for identity vector $\boldsymbol{I}$ to construct secret keys down the hierarchy. The algorithm takes in an identity vector $\boldsymbol{I}$, a secret key for that identity vector and an identity I_x. It outputs a secret key for the identity vector $\boldsymbol{I}||I_x$.

$\textbf{Decrypt}(\mathrm{CT}, \mathrm{SK}) \to M$ The decryption algorithm takes in a ciphertext CT for message M and for identity vector $\boldsymbol{I}_c$, and a secret key SK for identity vector $\boldsymbol{I}_k$. If $\boldsymbol{I}_k$ is a prefix of $\boldsymbol{I}_c$, it outputs the message M.

3.3 Spatial Encryption

In a spatial encryption scheme, the secret keys are associated to affine subspaces of $\mathbb{Z}_N^n$, and the delegation relation is defined by subspace inclusion.

Spatial encryption is a generalization of HIBE. Indeed, if we fix $n = D$ and consider an identity vector $\boldsymbol{I} = (I_1, \ldots, I_j)$, we can associate it in a bijective manner to the affine space $\{(I_1, \ldots, I_j, x_{j+1}, \ldots, x_n) | x_{j+1}, \ldots, x_n \in \mathbb{Z}_N\}$. The delegation relation is respected.

$\left.\begin{array}{l}\textbf{Setup}(1^\lambda) \to (\mathrm{PP}, \mathrm{MK}) \\ \textbf{KeyGen}(\mathrm{SK}_E, W) \to K \\ \textbf{Encrypt}(m, \boldsymbol{u}) \to \mathrm{CT}\end{array}\right\}$ have the same description than in the IBE scheme.

$\textbf{Delegate}(W_1, \mathrm{SK}_{W_1}, W_2) \to \mathrm{SK}_{W_2}$ The algorithm takes in an affine space W_1, a secret key for that space, and a subspace W_2 included in W_1. It outputs a secret key for W_2.

Decrypt$(\mathrm{CT}, \boldsymbol{u}, W, \mathrm{SK}_W) \to m$ The decryption algorithm takes in a ciphertext CT for message m and for vector $\boldsymbol{u}$, and a secret key SK for space W. If W contains $\boldsymbol{u}$, it first delegates SK_W to obtain $SK_{\boldsymbol{u}}$. Then it outputs the message m.

4 Our New Leakage-Resilient Spatial Encryption Scheme

We now present our construction of a leakage-resilient spatial encryption scheme with constant-size ciphertext, which can be seen as an extension of the leakage-resilient HIBE scheme in [LRW11]. The security proof of this system is available in Appendix A.

4.1 A Few Useful Notations

For a vector $v = (v_1, \ldots, v_n)^\top \in \mathbb{Z}_N^n$ of field elements, we use $g^{\boldsymbol{v}}$ to denote the vector of group elements

$$g^{\boldsymbol{v}} = (g^{v_1}, \ldots, g^{v_n})^\top \in \mathbb{G}^n$$

To simplify the notation, we introduce an operator

$$\begin{array}{rccc} \psi: & \mathbb{G}^n \times \mathbb{Z}_N^n & \to & \mathbb{G} \\ & (g^{\boldsymbol{v}}, w) & \mapsto & g^{\langle \boldsymbol{v}, \boldsymbol{w} \rangle} \end{array}$$

and its extension:

$$\begin{array}{rccc} \psi^*: & \mathbb{G}^n \times \mathbb{Z}_N^{n \times d} & \to & \mathbb{G}^d \\ & \left(g^{\boldsymbol{v}}, \left(\boldsymbol{C_1}, \ldots, \boldsymbol{C_d}\right)\right) & \mapsto & (\psi(g^{\boldsymbol{v}}, \boldsymbol{C_1}), \ldots, \psi(g^{\boldsymbol{v}}, \boldsymbol{C_d}))^\top \end{array}$$

We can notice that ψ is easily computable.

4.2 Construction

The system parameters for our spatial encryption system will be primes p_1, p_2, p_3 (where each $\log p_i$ is approximately the security parameter λ) and two groups $\mathbb{G}$ and $\mathbb{G}_T$ of order $N = p_1 p_2 p_3$, with a bilinear pairing $\hat{e}: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Additionally, the public parameters will include group elements $g_1 \in \mathbb{G}_{p_1}, g_3 \in \mathbb{G}_{p_3}, g_1^{\varphi} \in \mathbb{G}, t \in \mathbb{G}_T$ and a vector $g_1^{\boldsymbol{\alpha}} \in \mathbb{G}^n$.

A secret key for an affine space $W = \mathrm{Aff}(M, \boldsymbol{u})$ of dimension d will have the form

$$(\boldsymbol{k_\rho}, k_r, k_{\boldsymbol{u}}, \boldsymbol{k}_{\mathrm{del}}) = \left(g_1^{\boldsymbol{\rho}}, g_1^r, g_1^{\tau + r(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle}, g_1^{rM^\top \boldsymbol{\alpha}}\right) \times g_3^{\boldsymbol{\mu}} \in \mathbb{G}^{d+n+2}.$$

Setup(λ) generates the system parameters $N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T$. It then chooses parameters

$$g_1 \xleftarrow{R} \mathbb{G}_{p_1} \qquad g_3 \xleftarrow{R} \mathbb{G}_{p_3} \qquad \varphi \xleftarrow{R} \mathbb{Z}_N \qquad \boldsymbol{\alpha}, \boldsymbol{\rho}, \boldsymbol{\sigma} \xleftarrow{R} \mathbb{Z}_N^n \qquad \boldsymbol{\mu} \xleftarrow{R} \mathbb{Z}_N^{2n+2}$$

and secret parameter $\tau \xleftarrow{R} \mathbb{G}$, then computes $t = \hat{e}(g_1, g_1)^\tau$. It outputs public parameters

$$\mathrm{PP} = (N, g_1, g_3, g_1^\varphi, g_1^{\boldsymbol{\alpha}}, t, g_1^{\boldsymbol{\sigma}})$$

and secret key

$$\mathrm{SK}_E = \left(g_1^{\boldsymbol{\rho}}, g_1^r, g_1^{\tau + r\varphi - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle}, g_1^{r\boldsymbol{\alpha}}\right) \times g_3^{\boldsymbol{\mu}}.$$

Delegate$(W_1, \mathrm{SK}_{W_1}, W_2)$ takes two subspaces $W_1 = \mathrm{Aff}(M_1, \boldsymbol{u_1})$ and $W_2 = \mathrm{Aff}(M_2, \boldsymbol{u_2})$ and a key SK_{W_1} under the form $(\boldsymbol{k_\rho}, k_r, k_{\boldsymbol{u_1}}, \boldsymbol{k}_{\mathrm{del}})$. Let $\boldsymbol{\mu}$ be the $\mathbb{G}_{p_3}$ part of SK_{W_1}. We similarly divide $\boldsymbol{\mu}$ into $(\boldsymbol{\mu_\rho}, \mu_r, \mu_{\boldsymbol{u_1}}, \boldsymbol{\mu}_{\mathrm{del}})$. Since W_2 is a subspace of W_1, we must have $M_2 = M_1T$ and $\boldsymbol{u_2} = \boldsymbol{u_1} + M_1\boldsymbol{v}$ for some matrix T and vector $\boldsymbol{v}$. We can then compute a key for W_2:

$$\begin{aligned}
&\widehat{\mathrm{SK}}_{W_2} = (\boldsymbol{k_\rho}, k_r, \underbrace{k_{\boldsymbol{u_1}} \cdot \psi(\boldsymbol{k}_{\mathrm{del}}, v)}_{k_{\boldsymbol{u_2}}}, \underbrace{\psi^*(\boldsymbol{k}_{\mathrm{del}}, T)}_{\boldsymbol{k'}_{\mathrm{del}}}) \\
&= \left(g_1^{\boldsymbol{\rho}}, g_1^r, g_1^{\tau + r(\varphi + \langle \boldsymbol{u_1}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle} \cdot g_1^{r\boldsymbol{v}^\top {M_1}^\top \alpha}, g_1^{rT^\top {M_1}^\top \boldsymbol{\alpha}}\right) \times \left(g_3^{\boldsymbol{\mu_\rho}}, g_3^{\mu_r}, g_3^{\mu_{\boldsymbol{u_1}} + \langle \boldsymbol{v}, \boldsymbol{\mu}_{\mathrm{del}} \rangle}, g_3^{T^\top \boldsymbol{\mu}_{\mathrm{del}}}\right) \\
&= \left(g_1^{\boldsymbol{\rho}}, g_1^r, g_1^{\tau + r(\varphi + \langle \boldsymbol{u_1}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle + r\langle M_1\boldsymbol{v}, \boldsymbol{\alpha} \rangle}, g_1^{r{M_2}^\top \boldsymbol{\alpha}}\right) \times g_3^{\boldsymbol{\mu'}} \\
&= \left(g_1^{\boldsymbol{\rho}}, g_1^r, g_1^{\tau + r(\varphi + \langle \boldsymbol{u_2}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle}, g_1^{r{M_2}^\top \boldsymbol{\alpha}}\right) \times g_3^{\boldsymbol{\mu'}}
\end{aligned}$$

where $\boldsymbol{\mu'} = (\boldsymbol{\mu_\rho}, \mu_r, \mu_{\boldsymbol{u_1}} + \langle \boldsymbol{v}, \boldsymbol{\mu}_{\mathrm{del}} \rangle, T^\top \boldsymbol{\mu}_{\mathrm{del}})$.
Then we pick $\Delta r \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{\Delta\rho} \xleftarrow{R} \mathbb{Z}_p^n$ and $\boldsymbol{\Delta\mu} \xleftarrow{R} \mathbb{Z}_p^{d+n+2}$ to re-randomize it:

$$\begin{aligned}
\mathrm{SK}_{W_2} &= \hat{\mathrm{SK}}_{W_2} \times \left(g_1^{\boldsymbol{\Delta\rho}}, g_1^{\Delta r}, g_1^{\Delta r(\varphi + \langle \boldsymbol{u_2}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\Delta\rho}, \boldsymbol{\sigma} \rangle}, g_1^{\Delta r {M_2}^\top \boldsymbol{\alpha}}\right) \times g_3^{\boldsymbol{\Delta\mu}}. \\
&= \left(g_1^{\boldsymbol{\rho} + \boldsymbol{\Delta\rho}}, g_1^{r + \Delta r}, g_1^{\tau + (r + \Delta r)(\varphi + \langle \boldsymbol{u_2}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho} + \boldsymbol{\Delta\rho}, \boldsymbol{\sigma} \rangle}, g_1^{(r + \Delta r){M_2}^\top \boldsymbol{\alpha}}\right) \times g_3^{\boldsymbol{\mu'} + \boldsymbol{\Delta\mu}}.
\end{aligned}$$

Notice that **Delegate**$(W_1, \mathrm{SK}_{W_1}, W_1)$ is a re-randomization of SK_{W_1}.

Encrypt$(m, \boldsymbol{u})$, where m is encoded as an element of the target group $\mathbb{G}_T$, picks a random $s \xleftarrow{R} \mathbb{Z}_N^n$ and computes the ciphertext

$$\mathrm{CT} = (\mathrm{CT}^*, c_m) = (\boldsymbol{c_\sigma}, c_{\boldsymbol{u}}, c_s, c_m) = \left(g_1^{s\boldsymbol{\sigma}}, g_1^{-s(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle)}, g_1^s, m \cdot t^s\right) \in \mathbb{G}^{n+2} \times \mathbb{G}_T.$$

Decrypt$(\mathrm{CT}, \boldsymbol{u}, W, \mathrm{SK}_W)$ where $\mathrm{CT} = (\boldsymbol{c_\sigma}, c_{\boldsymbol{u}}, c_s, c_m)$ first delegates SK_W to obtain the key $\mathrm{SK}_{\boldsymbol{u}} = (\boldsymbol{k_\rho}, k_r, k_{\boldsymbol{u}}) = \left(g_1^{\boldsymbol{\rho}}, g_1^r, g_1^{\tau + r(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle}\right) \times g_3^{\boldsymbol{\mu}}$. It then recovers

$$\begin{aligned}
\frac{c_m}{\hat{e}_3(\mathrm{CT}^*, \mathrm{SK}_{\boldsymbol{u}})} &= \frac{c_m}{\hat{e}(\boldsymbol{c_\sigma}, \boldsymbol{k_\rho}) \cdot \hat{e}(c_{\boldsymbol{u}}, k_r) \cdot \hat{e}(c_s, k_{\boldsymbol{u}})} \\
&= \frac{m \cdot t^s}{\hat{e}(g_1, g_1)^{s\langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle - rs(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle) + s(\tau + r(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho}, \boldsymbol{\sigma} \rangle)}} \\
&= \frac{m \cdot \hat{e}(g_1, g_1)^{s\tau}}{\hat{e}(g_1, g_1)^{s\tau}} = m.
\end{aligned}$$

4.3 Security Definition

The security of our system is based on a game, called MasterLeakSpatial. In this game, the challenger first makes a call to **Setup**, to get a secret key for the whole space, and public parameters that it can give to the adversary. In a first phase, the adversary can make a polynomial number of queries, in any order:

- **Create** queries to store keys in an array;
- **Leak** queries to apply a leak function of his choice on any key (as long as the total number of leaked bits for a certain key does not exceed ℓ_{SK});
- or **Reveal** queries to recover a whole key (as long as it is not a key associated to the entire space, obviously).

At the end of this first phase, the adversary chooses a challenge vector $\boldsymbol{u}^*$ such that none of the revealed spaces (associated to its revealed keys) contain it, as well as two messages m_0, m_1. It gives them to the challenger, which chooses a random bit β and returns $\mathrm{CT}^* = \mathbf{Encrypt}(m_\beta, \boldsymbol{u}^*)$. From then on, the second phase starts, and the adversary can make **Create** and **Reveal** queries on keys whose space does not contain $\boldsymbol{u}^*$, naturally. It finally has to suggest a value for β. If it guesses right, it wins.

The scheme is considered leakage resilient if no PPT adversary can win the MasterLeakSpatial game with probability significantly better than 1/2. Indeed, roles are symmetric with respect to 1/2: if the adversary always loses the game, it is as concerning as if it had always won.

In order to formally define the MasterLeakSpatial security game, let us first introduce the following parameters:

- the set $\mathcal{R} \in \mathcal{P}(E)$ of spaces for which a key has been revealed;
- the array $\mathcal{T} \subset E \times \mathcal{SK} \times \mathbb{N}$ that holds tuples of the form (`space`, `key`, `leaked`). An example is shown in Figure 2.

	E	SK_E	0	W	SK_W	1	W'	$\mathrm{SK}_{W'}$	0	E	SK'_E	5	
		0			1			2			3		

Figure 2: Example of $\mathcal{T}$ array being filled in the MasterLeakSpatial game. Each cell contains the triplet (`space`, `key`, `leaked`).

The formal definition of the game MasterLeakSpatial, which is described in Figure 3, consists of the following phases:

Setup. The challenger makes a call to $\mathbf{Setup}(1^\lambda)$ and gets a key SK for the whole space E and the public parameters PP. It gives PP to the adversary.

Phase 1. In this phase, the adversary can make any of the following queries to the challenger, in any possible way: **Create**, **Leak**, **Reveal**, **Delegate**. These algorithms are listed in Figure 3.

Challenge. The adversary chooses a challenge vector $\boldsymbol{u}^*$ such that no element of $\mathcal{R}$ contains it, as well as two messages m_0, m_1 of equal size. It makes a call to $\mathbf{LR}(\boldsymbol{u}^*, m_0, m_1)$ and obtains a ciphertext CT^*.

Phase 2. This is the same as **Phase 1**, except the only allowed queries are **Create** and **Reveal** for secret keys whose space does not contain $\boldsymbol{u}^*$.

Guess. The adversary chooses a bit β' and calls $\mathbf{Finalize}(\beta')$. If the output is **True**, it succeeds.

Game MasterLeakSpatial

procedure Initialize
$(\mathrm{PP}, \mathrm{SK}) \xleftarrow{R} \mathsf{Setup}$
$\mathcal{R} \leftarrow \emptyset$
$\mathcal{T}[0] \leftarrow (E, \mathrm{SK}, 0)$
$\mathtt{nbKeys} \leftarrow 1$
$\beta \xleftarrow{R} \{0,1\}$
Return PP

procedure Create(h, W)
If $\mathcal{T}[h].\mathtt{space} = E$ then
  $\mathrm{SK}_W \xleftarrow{R} \mathsf{KeyGen}(\mathcal{T}[h].\mathtt{key}, W)$
  $\mathcal{T}[\mathtt{nbKeys}] \leftarrow (W, \mathrm{SK}_W, 0)$
  $\mathtt{nbKeys} \leftarrow \mathtt{nbKeys} + 1$
  Return SK_W
Else
  Return $\perp$

procedure Delegate(h, W')
$\mathrm{SK}_{W'} \leftarrow \mathsf{Delegate}(\mathcal{T}[h].\mathtt{space}, \mathcal{T}[h].\mathtt{key}, W')$
$\mathcal{T}[\mathtt{nbKeys}] \leftarrow (W', \mathrm{SK}_{W'}, 0)$
$\mathtt{nbKeys} \leftarrow \mathtt{nbKeys} + 1$
Return $\mathrm{SK}_{W'}$

procedure Leak(h, f)
$\mathrm{SK} \leftarrow \mathcal{T}[h].\mathtt{key}$
If $\mathcal{T}[h].\mathtt{leaked} + |f(\mathrm{SK})| \leq \ell_{\mathrm{SK}}$ then
  $\mathcal{T}[h].\mathtt{leaked} \leftarrow \mathcal{T}[h].\mathtt{leaked} + |f(\mathrm{SK})|$
  Return $f(\mathrm{SK})$
Else
  Return $\perp$

procedure Reveal(h)
If $\mathcal{T}[h].\mathtt{space} \neq E$ then
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{T}[h].\mathtt{space}$
  Return $\mathcal{T}[h].\mathtt{key}$
Else
  Return $\perp$

procedure LR$(\boldsymbol{u}^*, m_0, m_1)$
$\mathrm{CT}^* \xleftarrow{R} \mathsf{Encrypt}(\boldsymbol{u}^*, m_\beta)$
Return CT^*

procedure Finalize(β')
Return $(\beta' = \beta)$

Figure 3: Algorithms of the game MasterLeakSpatial.

The security definition we will use is the following:

Definition 1. *A spatial encryption system* Π *is* ℓ_{SK}*-master-leakage secure if for all PPT adversaries* $\mathcal{A}$ *it is true that*

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

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

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

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

4.4 Proof Overview

4.4.1 Dual System Encryption

The Dual System Encryption, first developed by Waters in [Wat09], is a methodology for proving security of encryption systems.

Both ciphertexts and secret keys will take on one of two indistinguishable forms: *normal*, or *semi-functional*. A semi-functional key cannot decrypt a semi-functional ciphertext. All other combinations work, as shown in Table 1.

This technique consists in gradually modifying the secret keys and ciphertexts of the system to make them *semi-functional*. When both secret keys and ciphertexts are semi-functional, proving security is straightforward.

Table 1: Different combinations of secret keys and ciphertexts.

secret key	ciphertext	decryption
normal	normal	correct (obviously)
normal	semi-functional	correct
semi-functional	normal	correct
semi-functional	semi-functional	incorrect

4.4.2 Hybrid Games

In order to prove that the advantage of the adversary in the MasterLeakSpatial game is negligible, we will define a sequence of games that are closely related to each other[2]. We will prove that the difference in probability of winning for the adversary is negligible between two consecutive games of this sequence, until we get to a game where both keys and ciphertexts will be semi-functional, so the adversary will not have any chance of winning the game.

First, we define MasterLeakSpatial*, which is the same game as MasterLeakSpatial except that all **Delegate**(h, W') calls have been replaced by **Create**$(0, W')$ calls. As the keys are identically distributed, the advantage of the adversary in MasterLeakSpatial is negligibly close to its advantage in MasterLeakSpatial*.

Then we define MasterLeakC, similar to MasterLeakSpatial* except that the challenge ciphertext is now semi-functional. Using Assumption 1, we will show that the advantages of the adversary in those games are negligibly close.

In MasterLeakCK, the semi-functional versions of the secret keys are stored in $\mathcal{T}$ as well: it contains quadruplets of the form (`space`, `key`, `keySF`, `leaked`). The **Create** calls use the normal versions of the keys while the **Leak** and **Reveal** queries apply to the

[2]The complete description of the algorithms of these games is available in Figures 5 and 6.

semi-functional versions. We will need Assumption 3 to prove that the advantage of the adversary is negligible in this game.

Finally, in the $\mathsf{MasterLeakCK}_j$ game, for the $j-1$ first keys the attacker creates, it will have access to the semi-functional versions of them via **Leak** or **Reveal** queries. For the other keys, it will have access to the normal versions (see also Figure 4). Notice that $\mathsf{MasterLeakCK}_0$ is exactly MasterLeakC while $\mathsf{MasterLeakCK}_{\texttt{nbKeys}}$ is exactly MasterLeakCK. We will show that if Assumption 2 holds, the difference of probability of winning for the adversary in $\mathsf{MasterLeakCK}_j$ versus $\mathsf{MasterLeakCK}_{j+1}$ is negligible, which will conclude the proof.

	W_0 $\widetilde{\mathrm{SK}}_{W_0}$ SK_{W_0} 4	W_1 $\widetilde{\mathrm{SK}}_{W_1}$ SK_{W_1} 6	W_2 $\widetilde{\mathrm{SK}}_{W_2}$ SK_{W_2} 2	W_3 $\widetilde{\mathrm{SK}}_{W_3}$ SK_{W_3} 3	
	$j-2$	$j-1$	j	$j+1$	

Figure 4: Example of $\mathcal{T}$ array being filled in the $\mathsf{MasterLeakCK}_j$ game. Each cell contains the quadruplet (`space`, `key`, `keySF`, `leaked`). The adversary has access to the semi-functional versions of the $j-1$ first keys, and to the normal versions of the other keys.

5 Concluding Remarks

We have designed the first leakage-resilient spatial encryption scheme, which is an important instance of generalized identity-based encryption. Though reasonably efficient, our scheme does not tolerate leakage during key updates. To address this problem and obtain a scheme with stronger security guarantees, a promising direction would be to adapt the result of [LLW11] to the spatial encryption scenario.

Game MasterLeakC

procedure Initialize
$(\mathrm{PP}, \mathrm{SK}) \xleftarrow{R} \mathsf{Setup}$
$\mathcal{R} \leftarrow \emptyset$
$\mathcal{T}[0] \leftarrow (E, \mathrm{SK}, 0)$
$\mathtt{nbKeys} \leftarrow 1$
$\beta \xleftarrow{R} \{0,1\}$
Return PP

procedure Create(h, W)
If $\mathcal{T}[h].\mathtt{space} = E$ then
  $\mathrm{SK}_W \xleftarrow{R} \mathsf{KeyGen}(\mathcal{T}[h].\mathtt{key}, W)$
  $\mathcal{T}[\mathtt{nbKeys}] \leftarrow (W, \mathrm{SK}_W, 0)$
  $\mathtt{nbKeys} \leftarrow \mathtt{nbKeys} + 1$
  Return SK_W
Else
  Return $\perp$

procedure Delegate(h, W')
Return **Create**$(0, W')$

procedure Leak(h, f)
$\mathrm{SK} \leftarrow \mathcal{T}[h].\mathtt{key}$
If $\mathcal{T}[h].\mathtt{leaked} + |f(\mathrm{SK})| \leq \ell_{\mathrm{SK}}$ then
  $\mathcal{T}[h].\mathtt{leaked} \leftarrow \mathcal{T}[h].\mathtt{leaked} + |f(\mathrm{SK})|$
  Return $f(\mathrm{SK})$
Else
  Return $\perp$

procedure Reveal(h)
If $\mathcal{T}[h].\mathtt{space} \neq E$ then
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{T}[h].\mathtt{space}$
  Return $\mathcal{T}[h].\mathtt{key}$
Else
  Return $\perp$

procedure LR$(\boldsymbol{u}^*, m_0, m_1)$
$\mathrm{CT}^* \xleftarrow{R} \boxed{\mathsf{EncryptSF}}(\boldsymbol{u}^*, m_\beta)$
Return CT^*

procedure Finalize(β')
Return $(\beta' = \beta)$

Game MasterLeakCK

procedure Initialize
$(\mathrm{PP}, \mathrm{SK}) \xleftarrow{R} \mathsf{Setup}$
$\boxed{\widetilde{\mathrm{SK}} \leftarrow \mathsf{KeyGenSF}(\mathrm{SK}, E)}$
$\mathcal{R} \leftarrow \emptyset$
$\mathcal{T}[0] \leftarrow (E, \mathrm{SK}, \boxed{\widetilde{\mathrm{SK}}}, 0)$
$\mathtt{nbKeys} \leftarrow 1$
$\beta \xleftarrow{R} \{0,1\}$
Return PP

procedure Create(h, W)
If $\mathcal{T}[h].\mathtt{space} = E$ then
  $\mathrm{SK}_W \xleftarrow{R} \mathsf{KeyGen}(\mathcal{T}[h].\mathtt{key}, W)$
  $\boxed{\widetilde{\mathrm{SK}}_W \xleftarrow{R} \mathsf{KeyGenSF}(\mathcal{T}[h].\mathtt{key}, W)}$
  $\mathcal{T}[\mathtt{nbKeys}] \leftarrow (W, \mathrm{SK}_W, \boxed{\widetilde{\mathrm{SK}}_W}, 0)$
  $\mathtt{nbKeys} \leftarrow \mathtt{nbKeys} + 1$
  Return $\widetilde{\mathrm{SK}}_W$
Else
  Return $\perp$

procedure Delegate(h, W')
Return **Create**$(0, W')$

procedure Leak(h, f)
$\widetilde{\mathrm{SK}} \leftarrow \boxed{\mathcal{T}[h].\mathtt{keySF}}$
If $\mathcal{T}[h].\mathtt{leaked} + |f(\widetilde{\mathrm{SK}})| \leq \ell_{\mathrm{SK}}$ then
  $\mathcal{T}[h].\mathtt{leaked} \leftarrow \mathcal{T}[h].\mathtt{leaked} + |f(\widetilde{\mathrm{SK}})|$
  Return $f(\widetilde{\mathrm{SK}})$
Else
  Return $\perp$

procedure Reveal(h)
If $\mathcal{T}[h].\mathtt{space} \neq E$ then
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{T}[h].\mathtt{space}$
  Return $\boxed{\mathcal{T}[h].\mathtt{keySF}}$
Else
  Return $\perp$

procedure LR$(\boldsymbol{u}^*, m_0, m_1)$
$\mathrm{CT}^* \xleftarrow{R} \mathsf{EncryptSF}(\boldsymbol{u}^*, m_\beta)$
Return CT^*

procedure Finalize(β')
Return $(\beta' = \beta)$

Figure 5: Algorithms of the games MasterLeakC and MasterLeakCK.

Game $\mathsf{MasterLeakCK}_j$

procedure Initialize

$(\mathrm{PP}, \mathrm{SK}) \xleftarrow{R} \mathsf{Setup}$
$\widetilde{\mathrm{SK}} \leftarrow \mathsf{KeyGenSF}(\mathrm{SK}, E)$
$\mathcal{R} \leftarrow \emptyset$
$\mathcal{T}[0] \leftarrow (E, \mathrm{SK}, \widetilde{\mathrm{SK}}, 0)$
$\mathtt{nbKeys} \leftarrow 1$
$\beta \xleftarrow{R} \{0,1\}$
Return PP

procedure Create(h, W)

If $\mathcal{T}[h].\mathtt{space} = E$ then
  $\mathrm{SK}_W \xleftarrow{R} \mathsf{KeyGen}(\mathcal{T}[h].\mathtt{key}, W)$
  $\widetilde{\mathrm{SK}}_W \xleftarrow{R} \mathsf{KeyGenSF}(\mathcal{T}[h].\mathtt{key}, W)$
  $\mathcal{T}[\mathtt{nbKeys}] \leftarrow (W, \mathrm{SK}_W, \widetilde{\mathrm{SK}}_W, 0)$
  $\mathtt{nbKeys} \leftarrow \mathtt{nbKeys} + 1$
  Return SK_W
Else
  Return $\perp$

procedure Delegate(h, W')

Return **Create**$(0, W')$

procedure Leak(h, f)

$\mathrm{SK} \leftarrow \boxed{\mathtt{nbKeys} > j}\ ?\ \mathcal{T}[h].\mathtt{key} : \mathcal{T}[h].\mathtt{keySF}$
If $\mathcal{T}[h].\mathtt{leaked} + |f(\mathrm{SK})| \leq \ell_{\mathrm{SK}}$ then
  $\mathcal{T}[h].\mathtt{leaked} \leftarrow \mathcal{T}[h].\mathtt{leaked} + |f(\mathrm{SK})|$
  Return $f(\mathrm{SK})$
Else
  Return $\perp$

procedure Reveal(h)

If $\mathcal{T}[h].\mathtt{space} \neq E$ then
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{T}[h].\mathtt{space}$
  Return $\boxed{\mathtt{nbKeys} > j}\ ?\ \mathcal{T}[h].\mathtt{key} : \mathcal{T}[h].\mathtt{keySF}$
Else
  Return $\perp$

procedure LR$(\boldsymbol{u}^*, m_0, m_1)$

$\mathrm{CT}^* \xleftarrow{R} \mathsf{EncryptSF}(\mathrm{PP}, \boldsymbol{u}^*, m_\beta)$
Return CT^*

procedure Finalize(β')

Return $(\beta' = \beta)$

Figure 6: Algorithms of the game $\mathsf{MasterLeakCK}_j$.

References

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, Berlin, Germany, March 15–17, 2009.

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer, Berlin, Germany.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.

[BH08] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 455–470, Melbourne, Australia, December 7–11, 2008. Springer, Berlin, Germany.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st FOCS*, pages 501–510. IEEE Computer Society Press, 2010.

[Bon07] Dan Boneh. Bilinear groups of composite order (invited talk). In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *PAIRING 2007*, volume 4575 of *LNCS*, page 1, Tokyo, Japan, July 2–4, 2007. Springer, Berlin, Germany.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 89–108, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.

[BWY11] Mihir Bellare, Brent Waters, and Scott Yilek. Identity-based encryption secure against selective opening attack. In Yuval Ishai, editor, *TCC 2011*,

volume 6597 of *LNCS*, pages 235–252, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st FOCS*, pages 511–520. IEEE Computer Society Press, 2010.

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *43rd ACM STOC*, pages 725–734. ACM Press, 2011.

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

[LRW11] Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 70–88, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479, Zurich, Switzerland, February 9–11, 2010. Springer, Berlin, Germany.

[MD11] Daisuke Moriyama and Hiroshi Doi. A fully secure spatial encryption scheme. *IEICE Transactions*, 94-A(1):28–35, 2011.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

A Security Proof

A.1 Semi-functionality

A *semi-functional* key has $\mathbb{G}_{p_2}$ parts, while a normal key has not. We pick at random g_2, a generator of $\mathbb{G}_{p_2}$.

KeyGenSF(SK, W) first calls **Delegate**(E, SK, W) so as to get a normal key $\mathrm{SK}_W = (\boldsymbol{k_\rho}, k_r, k_{\boldsymbol{u}}, \boldsymbol{k}_{\mathrm{del}})$. It then picks $\boldsymbol{\gamma} \xleftarrow{R} \mathbb{Z}_N^{n+2}$, $\boldsymbol{\theta} \xleftarrow{R} \mathbb{Z}_N^d$ and computes

$$\widetilde{\mathrm{SK}}_W = \left((\boldsymbol{k_\rho}, k_r, k_{\boldsymbol{u}}) \times g_2^{\boldsymbol{\gamma}}, \boldsymbol{k}_{\mathrm{del}} \times g_2^{\boldsymbol{\theta}} \right)$$

EncryptSF$(\boldsymbol{u}, m)$ first calls **Encrypt**$(\boldsymbol{u}, m)$ to get $\mathrm{CT} = (\boldsymbol{c_\sigma}, c_{\boldsymbol{u}}, c_s, c_m)$. It then picks $\boldsymbol{\delta} \xleftarrow{R} \mathbb{Z}_N^{n+2}$ and computes the ciphertext

$$\widetilde{\mathrm{CT}} = \left((\boldsymbol{c_\sigma}, c_{\boldsymbol{u}}, c_s) \times g_2^{\boldsymbol{\delta}}, c_m \right).$$

$\boldsymbol{\gamma}, \boldsymbol{\theta}, \boldsymbol{\delta}$ are called the *semi-functional parameters* of the secret key, the delegation key and the ciphertext, respectively. If someone uses a semi-functional key $\widetilde{\mathrm{SK}}$ for space $W = \mathrm{Aff}(M, \boldsymbol{u_1})$ with parameters $(\boldsymbol{\gamma}, \boldsymbol{\theta})$ to construct a secret key for vector $\boldsymbol{u_2} = \boldsymbol{u_1} + M\boldsymbol{v}$ with KeyGen, then this will be semi-functional with parameters $\boldsymbol{\gamma'} = \boldsymbol{\gamma} + (0, \ldots, 0, \langle \boldsymbol{v}, \boldsymbol{\theta} \rangle)$.

A semi-functional secret key for vector $\boldsymbol{u}$ is called *nominal* with respect to a ciphertext for vector $\boldsymbol{u'}$ if it can correctly decrypt it, thus if and only if:

$$\langle \boldsymbol{\gamma}, \boldsymbol{\delta} \rangle = 0 \bmod p_2 \text{ and } \boldsymbol{u} = \boldsymbol{u'}$$

because we get an extra term $\hat{e}(g_2, g_2)^{\langle \boldsymbol{\gamma}, \boldsymbol{\delta} \rangle}$ by the pairing. If the secret key cannot decrypt it, it is called *truly* semi-functional.

A.2 Lemma for Leakage Analysis

Our analysis of the leakage resilience of our system will rely on the following lemma from [BKKV10, LRW11], which is proven in [BFO08]. Below, we let $dist(X_1, X_2)$ denote the statistical distance of two random variables X_1 and X_2.

Lemma 1. *Let $m, l, d \in \mathbb{N}$, $m \geq l \geq 2d$ and let p be a prime. Let $X \xleftarrow{R} \mathbb{Z}_p^{m \times \ell}$, let $Y \xleftarrow{R} \mathbb{Z}_p^{m \times d}$, and let $T \xleftarrow{R} Rk_d(\mathbb{Z}_p^{\ell \times d})$, where $Rk_d(\mathbb{Z}_p^{\ell \times d})$ denotes the set of $\ell \times d$ matrices of tank d with entries in $\mathbb{Z}_p$. Let $f : \mathbb{Z}_p^{m \times d} \to W$ be some function. Then,*

$$dist((X, f(X \cdot T)), (X, f(Y))) \leq \varepsilon$$

as long as

$$|W| \leq 4 \cdot \left(1 - \frac{1}{p}\right) \cdot p^{\ell - (2d-1)} \cdot \varepsilon^2.$$

More precisely, we will use the following corollary.

Corollary 1. *Let $m \in \mathbb{N}, m \geq 3$, and let p be a prime. Let $\boldsymbol{\Delta} \xleftarrow{R} \mathbb{Z}_p^m$, $\boldsymbol{\nu} \xleftarrow{R} \mathbb{Z}_p^m$ and let $\boldsymbol{\nu'}$ be chosen uniformly randomly from the set of vectors in $\mathbb{Z}_p^m$ which are orthogonal to $\boldsymbol{\Delta}$ under the dot product modulo p. Let $f : \mathbb{Z}_p^m \to W$ be some function. Then,*

$$dist\left((\boldsymbol{\Delta}, f(\boldsymbol{\nu})), (\boldsymbol{\Delta}, f(\boldsymbol{\nu'}))\right) \leq \varepsilon$$

as long as

$$|W| \leq 4 \cdot \left(1 - \frac{1}{p}\right) \cdot p^{m-2} \cdot \varepsilon^2.$$

Proof. We apply Lemma 1 with $d = 1$ and $\ell = m - 1$. Y then corresponds to $\boldsymbol{\nu}$, while X corresponds to a basis of the orthogonal space of $\boldsymbol{\Delta}$. We note that $\boldsymbol{\nu'}$ is then distributed as $X \cdot T$, where $T \xleftarrow{R} Rk_1(\mathbb{Z}_p^{m-1\times 1})$. We note that X is determined by $\boldsymbol{\Delta}$, and is distributed as $X \xleftarrow{R} \mathbb{Z}_p^{m\times m-1}$, since $\boldsymbol{\Delta}$ is chosen uniformly randomly from $\mathbb{Z}_p^m$. It follows that

$$dist\left((\boldsymbol{\Delta}, f(\boldsymbol{\nu})), (\boldsymbol{\Delta}, f(\boldsymbol{\nu'}))\right) = dist((X, f(X \cdot T)), (X, f(Y))) \leq \varepsilon.$$

□

This corollary allows us to set $\ell_{\mathrm{SK}} = (n - 1 - 2c) \log p_2$ for our construction (we will have $n + 1 = m$), where c is any fixed positive constant (so that $\varepsilon = p_2^{-c}$ is negligible).

A.3 Proof

We will prove the following theorem.

Theorem 1. *Under assumptions 1, 2, 3 and for $\ell_{\mathrm{SK}} = (n - 1 - 2c) \log p_2$, where $c > 0$ is any fixed positive constant, our spatial encryption scheme is ℓ_{SK}-master-leakage secure.*

Table 2: Assumptions that will be used for the proofs.

game	game	assumption
MasterLeakSpatial	MasterLeakSpatial*	trivial
MasterLeakSpatial*	MasterLeakC	2.1
$\mathsf{MasterLeakCK}_j$	$\mathsf{MasterLeakCK}_{j+1}$	2.2
MasterLeakCK		2.3

Theorem 2. *Any polynomial-time attacker $\mathcal{A}$ has only a negligibly different probability of winning in MasterLeakSpatial versus MasterLeakSpatial*.*

Proof. It is easy to verify that the output of $\mathbf{Delegate}(W_1, \mathrm{SK}_{W_1}, W_2)$ is identically distributed to the output of $\mathbf{KeyGen}(\mathrm{SK}_E, W_2)$. □

Theorem 3. *If assumption 2.1 holds, any polynomial-time attacker $\mathcal{A}$ has only a negligibly different probability of winning in* MasterLeakSpatial* *versus* MasterLeakC.

Proof. We suppose there exists a PPT attacker $\mathcal{A}$ which attains a non-negligible difference in probability of winning between those two games. We will build a PPT algorithm $\mathcal{B}$ that breaks assumption 2.1 with non-negligible advantage.

$\mathcal{B}$ receives $\mathcal{D}_1 = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, g_3)$ and a challenge term $T \in \mathbb{G}_{p_1 p_2^?}$. Then it plays the MasterLeakSpatial* or the MasterLeakC game with $\mathcal{A}$ in the following way.

Setup. $\mathcal{B}$ picks $(\tau, \boldsymbol{\sigma}, \boldsymbol{\alpha}, \varphi) \xleftarrow{R} \mathbb{Z}_N \times \mathbb{Z}_N^n \times \mathbb{Z}_N^n \times \mathbb{Z}_N$ then gives

$$\mathrm{PP} = (N, g_1, g_3, g_1^{\varphi}, g_1^{\boldsymbol{\alpha}}, \hat{e}(g_1, g_1)^{\tau}, g_1^{\boldsymbol{\sigma}})$$

to $\mathcal{A}$, where N, g_1 and g_3 are given by the challenger.

Phase 1. Knowing $\boldsymbol{\alpha}$, the simulator can generate a secret key for the whole space E and use it to execute all secret queries (**Create**, **Leak**, **KeyGen**).

Challenge. The adversary $\mathcal{A}$ gives $\mathcal{B}$ two messages m_0 and m_1 and a challenge vector $\boldsymbol{u}^*$. The simulator $\mathcal{B}$ chooses $\beta \xleftarrow{R} \{0, 1\}$ and outputs the ciphertext

$$\mathrm{CT} = (\boldsymbol{c}_{\boldsymbol{\sigma}}, c_{\boldsymbol{u}^*}, c_s, c_m) = (T^{\boldsymbol{\sigma}}, T^{-(\varphi + \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle)}, T, m_\beta \cdot \hat{e}(T, g_1^{\tau})).$$

Phase 2. $\mathcal{B}$ works the same way as in **Phase 1**.

If $T = g_1^a g_2^b$, then the ciphertext is semi-functional. This implicitly sets

$$s = a \quad \text{and} \quad \boldsymbol{\delta} = b(\boldsymbol{\sigma}, -\varphi - \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle, 1).$$

s is properly distributed since $a \xleftarrow{R} \mathbb{Z}_N$ according to the assumption. $\boldsymbol{\delta}$ is properly distributed in the attacker's view because the factors $\sigma, -\varphi - \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle$ are only seen modulo p_1 in the public parameters and not modulo p_2. Thus they are random modulo p_2 in $\mathcal{A}$'s view. Therefore, $\mathcal{B}$ has properly simulated the MasterLeakC game.

If $T = g_1^a$, the ciphertext is normal since it has no $\mathbb{G}_{p_2}$ parts. In this case, $\mathcal{B}$ has properly simulated the MasterLeakSpatial* game. □

Theorem 4. *If assumption 2.2 holds, any polynomial-time attacker $\mathcal{A}$ has only a negligibly different probability of winning in* MasterLeakCK$_j$ *versus* MasterLeakCK$_{j+1}$.

Proof. $\mathcal{B}$ receives $\mathcal{D}_2 = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, g_3, g_1^x g_2^y, g_2^{y'}, g_3^{z'})$ and a challenge term $T \in \mathbb{G}_{p_1 p_2^? p_3}$. Then it plays the MasterLeakCK$_j$ or the MasterLeakCK$_{j+1}$ game with $\mathcal{A}$ as follows.

Setup. $\mathcal{B}$ picks $(\tau, \boldsymbol{\sigma}, \boldsymbol{\alpha}, \varphi) \xleftarrow{R} \mathbb{Z}_N \times \mathbb{Z}_N^n \times \mathbb{Z}_N^n \times \mathbb{Z}_N$ then gives

$$\mathrm{PP} = (N, g_1, g_3, g_1^{\varphi}, g_1^{\boldsymbol{\alpha}}, \hat{e}(g_1, g_1)^{\tau}, g_1^{\boldsymbol{\sigma}})$$

to $\mathcal{A}$, where N, g_1 and g_3 are given by the challenger.

Phase 1.

- For the first $j-1$ keys, $\mathcal{B}$ picks $(\boldsymbol{\rho'}, r, \boldsymbol{\mu}) \xleftarrow{R} \mathbb{Z}_N^n \times \mathbb{Z}_N \times \mathbb{Z}_N^{d+n+2}$ and, for the semi-functional parameters, $(\boldsymbol{\gamma'}, \boldsymbol{\theta'}) \xleftarrow{R} \mathbb{Z}_N^{n+2} \times \mathbb{Z}_N^d$. It then computes:

  $$\mathrm{SK} = \left(\left(g_1^{\boldsymbol{\rho'}}, g_1^r, g_1^{\tau + r(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle} \right) \times (g_2^{y'} g_3^{z'})^{\boldsymbol{\gamma'}}, g_1^{rM^{\top}\boldsymbol{\alpha}} \times (g_2^{y'} g_3^{z'})^{\boldsymbol{\theta'}} \right) \times g_3^{\boldsymbol{\mu}}$$

  The $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_3}$ parthosos are properly distributed and the semi-functional parameters are $\boldsymbol{\gamma} = y'\boldsymbol{\gamma'}$ and $\boldsymbol{\theta} = y'\boldsymbol{\theta'}$. These are properly distributed as well.

- For the j key, $\mathcal{B}$ will use the challenge term to generate the secret key. It picks $(\boldsymbol{\rho'}, \boldsymbol{\mu}) \xleftarrow{R} \mathbb{Z}_N^n \times \mathbb{Z}_N^{d+n+2}$ and outputs

  $$\mathrm{SK} = \left(T^{\boldsymbol{\rho'}}, T, g_1^{\tau} T^{\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle - \langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle}, T^{M^{\top}\boldsymbol{\alpha}} \right) \times g_3^{\boldsymbol{\mu}}$$

  If $T = g_1^a g_2^b g_3^c$, then this key is semi-functional with

  $$r' = a \qquad \boldsymbol{\rho} = a\boldsymbol{\rho'} \qquad \boldsymbol{\gamma} = b\left(\boldsymbol{\rho'}, 1, \varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle - \langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle\right) \qquad \boldsymbol{\theta} = bM^{\top}\boldsymbol{\alpha}.$$

  Again, since the public parameters only determine φ and $\boldsymbol{\alpha}$ modulo p_1, the semi-functional parameters are random modulo p_2 in $\mathcal{A}$'s view.

- From the $j+1$ key, $\mathcal{B}$ picks $(\boldsymbol{\rho'}, r, \boldsymbol{\mu}) \xleftarrow{R} \mathbb{Z}_N^n \times \mathbb{Z}_N \times \mathbb{Z}_N^{d+n+2}$ and computes:

  $$\mathrm{SK} = \left(g_1^{\boldsymbol{\rho'}}, g_1^r, g_1^{\tau + r(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle}, g_1^{rM^{\top}\boldsymbol{\alpha}} \right) \times g_3^{\boldsymbol{\mu}}$$

Challenge. In this phase, $\mathcal{B}$ has to create a semi-functional ciphertext with **EncryptSF**. $\mathcal{A}$ gives $\mathcal{B}$ two messages m_0 and m_1 and a challenge vector $\boldsymbol{u}^*$. $\mathcal{B}$ chooses $\beta \xleftarrow{R} \{0, 1\}$ and outputs the ciphertext

$$\widetilde{\mathrm{CT}} = (\boldsymbol{c}_{\boldsymbol{\sigma}}, c_{\boldsymbol{u}^*}, c_s, c_m) = \left((g_1^x g_2^y)^{\boldsymbol{\sigma}}, (g_1^x g_2^y)^{-(\varphi + \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle)}, (g_1^x g_2^y), m_\beta \cdot \hat{e}((g_1^x g_2^y), g_1^{\tau}) \right).$$

Phase 2. $\mathcal{B}$ works the same way as in **Phase 1**.

The ciphertext parameters are:

$$s = x \quad \text{and} \quad \boldsymbol{\delta} = y(\boldsymbol{\sigma}, -\varphi - \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle, 1).$$

s is properly distributed, but the semi-functional parameters are not: if the space of the j key contains the challenge vector $\boldsymbol{u}^*$, the secret key is nominal with respect to the ciphertext. Indeed:

$$\begin{aligned}
& (\boldsymbol{\gamma} + (0, \ldots, 0, \langle \boldsymbol{v}, \boldsymbol{\theta} \rangle)) \cdot \boldsymbol{\delta} \\
= \; & b \left(\boldsymbol{\rho'}, 1, \varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle - \langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle + \left\langle \boldsymbol{v}, M^\top \boldsymbol{\alpha} \right\rangle \right) \cdot y(\boldsymbol{\sigma}, -\varphi - \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle, 1) \\
= \; & by \left(\langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle - \varphi - \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle + \varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle - \langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle + \langle M\boldsymbol{v}, \boldsymbol{\alpha} \rangle \right) \\
= \; & by \langle \boldsymbol{u} + M\boldsymbol{v} - \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle \\
= \; & 0 \pmod{p_2}
\end{aligned}$$

Notice that if the space of the j key contains a vector $\boldsymbol{w}$ which is equal to $\boldsymbol{u}^*$ modulo p_2, we obtain the same result. With the help of two lemmas, we will show that the change in the adversary's advantage is negligible.

Lemma 2. *If assumption 2 holds, then for any PPT adversary $\mathcal{A}$, $\mathcal{A}$'s advantage in the $\mathsf{MasterLeakC}_j$ game, or in the $\mathsf{MasterLeakC}_{j+1}$ game, changes only by a negligible amount if we restrict it to make queries only on the challenge vector and on spaces such that no component of any vector of them is equal to a respective component from the challenge vector $\boldsymbol{u}^*$ modulo p_2.*

Proof. If there exists an adversary whose advantage changes by a non-negligible amount under this restriction, we can find a non-trivial factor of N with non-negligible probability. So we can break Assumption 2, using the same proof as [LW10]).

The simulator plays $\mathsf{MasterLeakC}_j$ or $\mathsf{MasterLeakC}_{j+1}$, using the terms from Assumption 2 to create the semi-functional keys and ciphertext. It worlds in a way similar to the simulator in the reduction shown in Theorem 3. □

Lemma 3. *We suppose the leakage is at most $\ell_{\mathrm{SK}} = (n - 1 - 2c) \log p_2$, where $c > 0$ is a positive constant. Then, for any PPT adversary $\mathcal{A}$, $\mathcal{A}$'s advantage in the $\mathsf{MasterLeakC}_{j+1}$ game changes only by a negligible amount when the truly semi-functional j key is replaced by a nominal semi-functional key whenever $\mathcal{A}$ declares the j key to be associated to a space that contains the challenge ciphertext vector.*

Proof. Let us suppose there exists a PPT algorithm $\mathcal{A}$ whose advantage changes by a non-negligible amount when the $\mathsf{MasterLeakC}_{j+1}$ game changes as described above. Using $\mathcal{A}$, we will create a PPT algorithm $\mathcal{B}$ which will distinguish between $(\boldsymbol{\Delta}, f(\boldsymbol{\nu}))$ and $(\boldsymbol{\Delta}, f(\boldsymbol{\nu'}))$

from Corollary 1 with non-negligible advantage (when $m = n + 1$ and $p = p_2$). This will yield a contradiction, since these distributions have a negligible statistical distance.

$\mathcal{B}$ simulates $\mathsf{MasterLeakC}_{j+1}$ as follows. It runs **Setup** and gives $\mathcal{A}$ the public parameters. Since $\mathcal{B}$ knows $\boldsymbol{\alpha}$ and generators of all the subgroups, it can respond to A's queries in **Phase 1**.

With non-negligible probability, the j key $\mathcal{A}$ chooses in **Phase 1** must be associated to a space that contains $\boldsymbol{u}^*$. (If it only did this with negligible probability, then the difference in advantages whenever it happens would be negligible.)

$\mathcal{B}$ will not create that key but instead will encode the leakage $\mathcal{A}$ asks for, as a single polynomial-time computable function f with domain $\mathbb{Z}_{p_2}^{n+1}$ and with an image of size $2^{\ell_{\mathrm{SK}}}$. $\mathcal{B}$ receives a sample $(\boldsymbol{\Delta}, f(\boldsymbol{\Gamma}))$, where $\boldsymbol{\Gamma}$ is either distributed as $\boldsymbol{\nu}$ or as $\boldsymbol{\nu'}$. $\mathcal{B}$ will use $f(\Gamma)$ to answer all of $\mathcal{A}$'s leakage queries on the j key by implicitly defining this key as follows.

$\mathcal{B}$ chooses $(r_1, r_2, \boldsymbol{\theta}) \in \mathbb{Z}_{p_2}^{d+2}$. We let g_2 denote a generator of $\mathbb{G}_{p_2}$. $\mathcal{B}$ implicitly sets the $\mathbb{G}_{p_2}$ components of the key to be $g_2^{\boldsymbol{\Gamma'}}$, where $\boldsymbol{\Gamma'}$ is defined to be:

$$\boldsymbol{\Gamma'} = (\Gamma_1, \ldots, \Gamma_n, \Gamma_{n+1} + r_1, r_2, \boldsymbol{\theta}).$$

$\mathcal{B}$ defines the other components of the key to fit their appropriate distribution.

At some point, $\mathcal{A}$ declares the challenge vector $\boldsymbol{u}^*$. If the space of the j key does not contain $\boldsymbol{u}^*$, then $\mathcal{B}$ aborts the simulation and guesses whether $\boldsymbol{\Gamma}$ is orthogonal to $\boldsymbol{\Delta}$ randomly. However, the simulation continues with non-negligible probability. Suppose the space of the j key is $W = \mathrm{Aff}(M, \boldsymbol{u})$ and let $\boldsymbol{v}$ be the vector such that $\boldsymbol{u}^* = \boldsymbol{u} + M\boldsymbol{v}$.

$\mathcal{B}$ chooses a random element $t_2 \in \mathbb{Z}_{p_2}$ such that $\Delta_{n+1} r_1 + (r_2 + \langle \boldsymbol{v}, \boldsymbol{\theta} \rangle) t_2 = 0 \pmod{p_2}$. It then constructs the challenge ciphertext, using $(\boldsymbol{\Delta}, t_2) \in \mathbb{Z}_{p_2}^{n+2}$ as parameter.

If $\boldsymbol{\Gamma}$ is orthogonal to $\boldsymbol{\Delta}$, then the j key is nominally semi-functional (and well distributed):

$$\begin{aligned} ((\Gamma_1, \ldots, \Gamma_n, \Gamma_{n+1} + r_1, r_2) + (0, \ldots, 0, \langle \boldsymbol{v}, \boldsymbol{\theta} \rangle)) \cdot \boldsymbol{\Delta} &= \boldsymbol{\Gamma} \cdot \boldsymbol{\Delta} + \Delta_{n+1} r_1 + (r_2 + \langle \boldsymbol{v}, \boldsymbol{\theta} \rangle) t_2 \\ &= 0 \pmod{p_2}. \end{aligned}$$

If $\boldsymbol{\Gamma}$ is not orthogonal to $\boldsymbol{\Delta}$, then the challenge key is truly semi-functional (and also well distributed).

It is clear that $\mathcal{B}$ can easily handle **Phase 2** queries, since the j key cannot be queried when its space contains $\boldsymbol{u}^*$. Therefore, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain a non-negligible advantage in distinguishing $(\boldsymbol{\Delta}, f(\boldsymbol{\nu}))$ and $(\boldsymbol{\Delta}, f(\boldsymbol{\nu'}))$, which violates Corollary 1. □

The above lemmas conclude the theorem.

□

Theorem 5. *If assumption 3 holds, any polynomial-time attacker $\mathcal{A}$ has only a negligible advantage in* MasterLeakCK.

Proof. We suppose there exists a PPT attacker $\mathcal{A}$ which attains a non-negligible advantage in MasterLeakCK. We will build a PPT algorithm $\mathcal{B}$ that breaks assumption 3 with non-negligible advantage.

$\mathcal{B}$ receives $\mathcal{D}_3 = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_1, g_2, g_3, g_1^\tau g_2^y, g_1^{x'} g_2^{y'})$ and a challenge term T which is either $\hat{e}(g_1, g_1)^{\tau x'}$ or a random term of $\mathbb{G}_T$. Algorithm $\mathcal{B}$ works as follows.

Setup. $\mathcal{B}$ picks $(\boldsymbol{\sigma}, \boldsymbol{\alpha}, \varphi) \xleftarrow{R} \mathbb{Z}_N^n \times \mathbb{Z}_N^n \times \mathbb{Z}_N$. Notice that now τ is unknown. The term $\hat{e}(g_1, g_1)^\tau$ is computed as $\hat{e}(g_1^\tau g_2^y, g_1)$. It gives $\mathrm{PP} = (N, g_1, g_3, g_1^\varphi, g_1^{\boldsymbol{\alpha}}, \hat{e}(g_1, g_1)^\tau, g_1^{\boldsymbol{\sigma}})$ to $\mathcal{A}$, where N, g_1 and g_3 are given by the challenger.

Phase 1. For each secret key requested by the adversary, the simulator $\mathcal{B}$ picks the random exponents $(r, \boldsymbol{\rho}, \boldsymbol{\mu}) \xleftarrow{R} \mathbb{Z}_N \times \mathbb{Z}_N^n \times \mathbb{Z}_N^{d+n+2}$ and for the semi-functional parameters, $\boldsymbol{\gamma'} \xleftarrow{R} \mathbb{Z}_N^{n+2}$ and $\boldsymbol{\theta} \xleftarrow{R} \mathbb{Z}_N^d$. It uses the secret key

$$\mathrm{SK} = \left(\left(g_1^{\boldsymbol{\rho}}, g_1^r, (g_1^\tau g_2^y) g_1^{r(\varphi + \langle \boldsymbol{u}, \boldsymbol{\alpha} \rangle) - \langle \boldsymbol{\rho'}, \boldsymbol{\sigma} \rangle} \right) \times g_2^{\boldsymbol{\gamma'}}, g_1^{rM^\top \boldsymbol{\alpha}} \times g_2^{\boldsymbol{\theta}} \right) \times g_3^{\boldsymbol{\mu}}$$

It is a properly distributed semi-functional key with parameters

$$\boldsymbol{\gamma} = \boldsymbol{\gamma'} + (0, \ldots, 0, y) \quad \text{and} \quad \boldsymbol{\theta}.$$

Challenge. The adversary $\mathcal{A}$ gives $\mathcal{B}$ two messages m_0, m_1 and a challenge vector $\boldsymbol{u}^*$. The simulator chooses $\beta \xleftarrow{R} \{0, 1\}$ and outputs the following ciphertext:

$$\mathrm{CT} = \left((g_1^{x'} g_2^{y'})^{\boldsymbol{\sigma}}, (g_1^{x'} g_2^{y'})^{-\varphi - \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle}, (g_1^{x'} g_2^{y'}), m_\beta \cdot T \right).$$

Phase 2. $\mathcal{B}$ works the same way as in **Phase 1**.

If $T = \hat{e}(g_1, g_1)^{\tau x'}$, then we get a semi-functional ciphertext of m_β with parameters

$$s = x' \quad \text{and} \quad \boldsymbol{\delta} = y'(\boldsymbol{\sigma}, -\varphi - \langle \boldsymbol{u}^*, \boldsymbol{\alpha} \rangle, 1).$$

$\boldsymbol{\delta}$ is properly distributed since all terms are random modulo p_2. Therefore, $\mathcal{B}$ has properly simulated game MasterLeakCK.

If $T \xleftarrow{R} \mathbb{G}_T$, then c_m is entirely random and we get a semi-functional ciphertext of a random message. Hence, the value of β is information-theoretically hidden and the probability of success of $\mathcal{A}$ is exactly $1/2$, since $\beta \xleftarrow{R} \{0, 1\}$. Therefore, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 3 with non-negligible advantage. □

This concludes the proof of Theorem 1.