Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage

Zvika Brakerski
Weizmann Institute of Science
zvika.brakerski@weizmann.ac.il

Yael Tauman Kalai
Microsoft Research
yael@microsoft.com

Jonathan Katz
University of Maryland
jkatz@cs.umd.edu

Vinod Vaikuntanathan
Microsoft Research
vinodv@alum.mit.edu

***Abstract*—In recent years, there has been a major effort to design cryptographic schemes that remain secure even when arbitrary information about the secret key is leaked (e.g., via side-channel attacks). We explore the possibility of achieving security under *continual* leakage from the *entire* secret key by designing schemes in which the secret key is updated over time. In this model, we construct public-key encryption schemes, digital signatures, and identity-based encryption schemes that remain secure even if an attacker can leak a constant fraction of the secret memory (including the secret key) in each time period between key updates. We also consider attackers who may probe the secret memory during the updates themselves. We stress that we allow unrestricted leakage, without the assumption that "only computation leaks information". Prior to this work, constructions of public-key encryption schemes secure under continual leakage were not known even under this assumption.**

I. Introduction

In most of cryptography, secrecy of secret keys is paramount, and compromise of even a portion of the secret key yields a break of the entire cryptosystem. In practice, unfortunately, it is difficult to maintain secrecy of cryptographic keys. For one, it has been shown that various *side-channel attacks* [27], [28], exploiting physical characteristics of the execution of a cryptographic algorithm (e.g., timing measurements, power consumption, or electromagnetic radiation), can be used to obtain information about secret keys. More recently, so-called "cold boot" attacks [21] have been used to recover some (noisy) fraction of the secret keys used in poor cryptographic implementations.

Responding to this challenge, cryptographers over the past several years have begun to investigate the construction of cryptographic schemes that are provably resilient to various forms of key leakage. There are, of course, many different ways in which this desired notion of *leakage resilience* can be modeled. Early work (e.g., [35], [10]) focused on an adversary who could learn (a bounded number of) *specific bits* of the secret key. Ishai, et al. [23], [22] considered attackers who could probe the values of specific wires in a circuit implementing the functionality of interest. Petit, et al. [33] construct pseudorandom generators that are secure against specific, naturally occurring, classes of leakage such as the Hamming weight leakage.

Micali and Reyzin [30] initiated work on a general model of leakage in which the adversary can learn arbitrary information about portions of memory that are accessed during a particular computational step, but is assumed to learn nothing about any portions of memory that are untouched.

Dziembowski and Pietrzak [17], [34] construct stream ciphers in this model, in which the secret key is updated over time [4] and the adversary can leak a bounded amount of information between key updates (subject to the limitation just mentioned) but an *unbounded* amount of information overall. Faust, et al. [18] construct a signature scheme that is secure in this leakage model.

We emphasize that in this leakage model, the leakage obtained by the adversary at any time period may depend *only* on those portions of the secret key that are actively used in the computation during that period; inactive portions of the key cannot be probed. This allows for "shielding" parts of the secret key; in particular, if in every time period some (different) part of the secret is shielded, then the shielded portion of the key is uncorrelated with the leakage that occurred in that time period; that portion of the key can thus be used to "refresh" the rest of the secret key.

The above model, in which "only computation leaks information", fails to capture the cold boot attack discussed earlier, as well as other attacks in which information leaks from portions of memory not directly accessed during a particular computation. This motivates consideration of a more general model in which the adversary can learn an arbitrary function of *all* the secret information. (Depending on the exact details of the model, the leakage may depend on the secret key only, or may depend also on other secret information such as random coins used by the algorithm.) Of course, *some* restriction must be placed on the leaked information or else the adversary could simply leak the entire secret key itself!

A complementary line of work, initiated by Akavia, Goldwasser, and Vikuntanathan [1], restricts the *total-length* of the leaked information, however, allows the leakage to depend on *all* parts of the memory arbitrarily. Namely, they impose a *global* bound on the amount of leakage in the entire lifetime of the scheme, *regardless of the amount of computation performed*. In particular, the leakage needs to be shorter than the secret key. Variants of this so called *bounded memory leakage model* require that the secret key retain sufficient min-entropy given the leakage [31] (see also [15]), or require that the secret key remain computationally hard to compute given the leakage [14]. Subsequent work in the bounded memory leakage model includes [25], [3], [12], [2].

A. Continual Memory Leakage

In this work, we introduce a new model that retains the strongest features of both the models above. Specifically, just as in the "leaky computation" model we consider schemes in which the secret key is updated over time and the total leakage over the lifetime of the system is *unbounded*. As in the bounded-leakage model, though, we allow the leakage between key updates to be an *arbitrary* function of the secret state, subject only to a restriction on the total length of the leakage.[1] (In particular, we no longer make the assumption that "only computation leaks information".) We refer to this as the model of *continual memory leakage*. Our model addresses deficiencies of both prior models in that the adversary we consider is unrestricted in both *time* (since leakage may occur forever) and *space* (since leakage may always depend on all portions of the secret memory). We remark, however, that we do make the implicit assumption in our model that data can be completely erased from memory (as otherwise the adversary would be able, over time, to leak the entire initial secret key).

For concreteness, we illustrate the model of continual memory leakage in the context of digital signature schemes. Initially, a public and secret key-pair is generated and the secret key is stored on some signing device. The lifetime of the system is divided into discrete time periods, where during each time period the signing device may issue multiple signatures. At the end of each time period a "refresh" or "update" operation is performed on the secret key, and the old secret key is erased. We stress that, as in forward-secure cryptosystems, the public key remains fixed throughout the lifetime of the system and is all that is needed for verification.

We consider an attacker that, as usual, can ask for signatures of any messages of its choice at any time. The attacker can also leak information about the secret state of the signing device, as we now make precise. We view the signing device as containing two types of memory:

1) *Public memory* that stores, e.g., the public verification key and any public randomness used.
2) *Secret memory* that stores the secret key and any secret randomness used for signature computation.

We assume the attacker can see the contents of the public memory in its entirety, at all times. Leakage from the secret memory is bounded *per time period*, but unbounded overall.

Formally, at each time period, the adversary can specify a leakage function f, and is given $f(sk)$, where sk is the secret key at that time period. In addition, each time the adversary requests a signature we also allow him to specify a leakage function f; in addition to the signature, the adversary is given the result of f applied to the entire secret state at that point in time (including the randomness used for the signature computation). The only restriction placed on f is that the output length of all such leakage functions during any specific time period should be bounded to some pre-specified fraction of the number of bits of the secret state.

In addition to the above, which treats leakage during the signing process, our model also incorporates leakage that may occur during the key-update process itself.

B. Our Results

In addition to formalizing the model of continual memory leakage, we also show constructions of public-key encryption schemes, digital signatures, and identity-based encryption (IBE) schemes that are secure in this setting.

The cornerstone of our results is a public-key encryption scheme that is secure against continual memory leakage. (We remark that prior to our work, there were no known public-key encryption schemes resilient to continual leakage even under the assumption that "only computation leaks information".) We show two constructions: The first is secure even if a $(1/2 - o(1))$ fraction of the secret state is leaked per time period; its security is based on the decisional linear assumption in groups with a bilinear map. A variant relies on the less standard SXDH assumption[2] and is secure even if a $(1 - o(1))$ fraction of the secret state is leaked per time period. Both schemes can also be shown resilient to $O(\log k)$ bits of leakage during the key-update process itself, where k is the security parameter. (If we are willing to rely on sub-exponential hardness assumptions, we can tolerate $O(k^\epsilon)$ bits of leakage from each key update.) We note that while tolerating a logarithmic-length leakage *once* is easy and can be done generically (since the leakage can be guessed), tolerating logarithmic-length leakage *repeatedly* is much more difficult.

It is easy to see that any public-key encryption scheme resilient to continual leakage automatically implies a private-key encryption scheme with similar leakage resilience. Thus, our scheme can be used to enable two parties who share a secret key to interact over an insecure channel in the presence of side-channel attacks. The parties simply *individually and independently* update their secret keys. Thus, at any point of time, the two communicating parties might have completely different secret keys, and still they will be able to communicate meaningfully. We stress that, as opposed to previous solutions to this problem, here the key update requires no interaction whatsoever!

Our public-key encryption scheme is obtained by suitably modifying the identity-based encryption scheme due to Brakerski and Kalai [9] (for which no leakage resilience was claimed). We can extend our construction to obtain an IBE scheme secure against continual memory leakage of users' individual secret keys. Achieving resilience to leakage from

[1] One can consider more general bounded-leakage models between updates. Indeed, our results extend to the more general "leftover entropy" model of Naor and Segev [31], which requires only that the secret key has sufficient min-entropy conditioned on the leakage.

[2] If $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear maps, the SXDH assumption is that the decisional Diffie-Hellman problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

the master secret key remains an interesting open problem (in any of the models for leakage resilience).

Finally, we show a generic technique (following Katz and Vaikuntanathan [25]) that transforms any public-key encryption scheme resilient to continual leakage to a signature scheme secure in the same model. The basic transformation results in a signature scheme that tolerates leakage from the secret key but not from the entire secret state (i.e., not from the randomness used by the signing procedure). Under additional cryptographic assumptions (most notably, the existence of short non-interactive arguments, such as cs-proofs [29]), we show how leakage from the entire secret state can be tolerated.

C. Other Prior and Concurrent Work

Besides the related work mentioned already, there has also recently been research aimed at achieving leakage resilience by relying on a small amount of perfectly *leakage-proof hardware*. In this model, Juma and Vahlis [24] and Goldwasser and Rothblum [20] show how to obtain continual leakage resilience for general cryptographic algorithms, based also on the assumption that "only computation leaks information" (for computation not done on the secure hardware). Faust, et al. [19] show how to use leakage-proof hardware to tolerate length-bounded leakage computed by a function in $\mathbf{AC}^0$

In work done concurrently with our own, Dodis, et al. [13] construct efficient signature schemes, identification schemes, and authenticated key-agreement protocols that are secure in the model of continual memory leakage. Unlike this work, they do not address the issue of leakage from the key-update process and, furthermore, they do not construct public-key encryption schemes or IBE schemes.

D. Paper Organization

Due to space constraints, this extended abstract only contains an overview of our results. We refer the interested reader to the full version [8] for further details and proofs.

In Section II we present our model of continual memory leakage, and give a formal definition only for the case of public-key encryption. (Definitions for IBE and signature schemes can be found in the full version of this work.) In Section III we introduce a linear-algebraic theorem that we use; the theorem shows (roughly) that random subspaces are leakage resilient (in a way we make precise there). We describe our public-key encryption scheme in Section IV, our identity based encryption scheme in Section V, and our signature scheme in Section VI.

II. A Model for Continual Memory Leakage

In this section, we formalize our model of *continual memory leakage* (i.e., the CML model). We start by describing the model at a high level in the context of signature schemes, and then, in the following section, we provide a formal definition for the case of public-key encryption. (The formal definition for signature schemes is complicated by the fact that the signing may be randomized, and thus we need to deal with leakage of the randomness used during the signing process; in contrast, decryption is deterministic.)

In the CML model, the lifetime of the system is divided into discrete time intervals, and the secret key is updated at the end of every such interval. Within a given interval, the adversary may submit a *leakage function* f of his choice, and is given $f(sk)$. In addition, as usual, the adversary may make signing queries to obtain valid signatures on messages of its choice. At the time of each signing query, though, the adversary is also allowed to submit a *leakage function* f; in return, the adversary is given $f(sk, r)$ where sk is the secret key (at the current time interval) and r is the secret randomness used to answer the signing query. (We assume r is securely erased once signature generation is complete.) The signing process may also use some "public randomness", which is given to the adversary "for free" and not counted toward the leakage bound. We restrict the leakage (i.e., the output length of f) per signature query to be some fraction of the total length (i.e., $|sk| + |r|$) of the secret state. We also restrict the total leakage per time interval to be some fraction of the length of the secret key.

At the end of an interval, the secret key is updated and the old secret key is erased. We allow the adversary also to probe the memory during this update process. The leakage during the key-update procedure is counted toward both the previous interval and the next interval; this is because the secret keys for both time intervals are completely determined by the inputs to the key-update procedure.

In the formulation of the model just described, the leakage is restricted to be strictly smaller than the length of the secret key (following [1]). Other restrictions could also be used; for example, we could require only that the secret key has sufficient min-entropy given the leakage (as in [31]), or that the secret key should be hard to compute given the leakage (see [14]). We choose the first restriction for simplicity, but note that our proofs go through as long as the min-entropy requirement is satisfied. On the other hand, we do not know how to prove our results using the notion of [14] and leave open the question of constructing schemes secure with respect to that notion.

A. CML-*Secure Public-Key Encryption*

We now give a formal definition of public-key encryption in the CML model. An encryption scheme consists of the following algorithms which run in time polynomial in k:

- *Key generation.* Gen takes as input a security parameter 1^k and outputs a secret key sk and a public key pk. We denote this by $(sk, pk) \leftarrow \mathsf{Gen}(1^k)$.
- *Encryption.* Enc takes a public key pk and a message m, and outputs a ciphertext c. We write $c \leftarrow \mathsf{Enc}_{pk}(m)$.
- *Decryption.* Dec takes keys sk, pk and a ciphertext c and outputs a message m'. We write $m' := \mathsf{Dec}_{sk,pk}(c)$.

- *Key update.* Update Takes as input keys sk, pk and outputs a "refreshed" secret key sk'. We denote this by $sk' \leftarrow \mathsf{Update}_{pk}(sk)$.

We remark that because our algorithms are required to run in time polynomial in k, we must not allow the secret key to grow too large following an update. In our construction, $|sk| = |\mathsf{Update}_{pk}(sk)|$ (i.e. the size of the secret key remains unchanged by update operations).

We require that for all m and all polynomially bounded $t \in \mathbb{N}$, setting $(sk_0, pk) \leftarrow \mathsf{Gen}(1^k)$, $sk_i \leftarrow \mathsf{Update}_{pk}(sk_{i-1})$ for $i \in [t]$, and $c \leftarrow \mathsf{Enc}_{pk}(m)$, we have $m = \mathsf{Dec}_{sk_t,pk}(c)$ with all but negligible probability (where the probability is over all randomness in the experiment).

We next define semantic security (i.e., security against chosen-plaintext attacks) in the CML model. Our definition has three leakage parameters ρ_G, ρ_U, ρ_M, where ρ_G bounds the leakage rate from the key-generation process, ρ_U bounds the leakage rate from the update process, and ρ_M is a "global" leakage bound that is enforced between key updates. Taking $\rho_G = \rho_U = 0$ corresponds to allowing leakage only during "normal" operation of the system, and not from the key-generation or key-update processes.

Definition 1. *Encryption scheme* (Gen, Enc, Dec, Update) *is* CML-secure with leakage rate (ρ_G, ρ_U, ρ_M) *if the advantage of any* PPT *adversary $\mathcal{A}$ in the following game is negligible:*

1) Initialize. *$\mathcal{A}$ specifies a circuit f with $|f(r,\tau)| \leq \rho_G \cdot |r|$ for all r, τ. The challenger chooses "secret randomness" r and "public randomness" τ, generates $(sk_0, pk) \leftarrow \mathsf{Gen}(1^k; r, \tau)$, sends $(pk, \tau, f(r,\tau))$ to the adversary, and sets $i{:=}0$ and $L_0{:=}|f(r,\tau)|$.*
2) Leakage and updates. *$\mathcal{A}$ makes the following queries:*
   - *Update queries* (update, f)*, where f is a circuit with $|f(sk, r, \tau)| \leq \rho_U \cdot (|sk| + |r|)$ for all sk, r, τ. The challenger chooses "secret randomness" r and "public randomness" τ, and computes $sk_{i+1} := \mathsf{Update}_{pk}(sk_i; r, \tau)$. If $L_i + |f(sk_i, r, \tau)| \leq \rho_M \cdot |sk_i|$ then the challenger returns $(\tau, f(sk_i, r, \tau))$ to the adversary, sets $i := i + 1$, and sets $L_{i+1} := |f(sk_i, r, \tau)|$. Otherwise, the challenger aborts.*
   - *Leakage queries* (leak, f)*, where f is a circuit. If $L_i + |f(sk_i)| \leq \rho_M \cdot |sk_i|$ then the challenger returns $f(sk_i)$ to the adversary and sets $L_i := L_i + |f(sk_i)|$. Otherwise, the challenger aborts.*
3) Challenge. *$\mathcal{A}$ outputs two messages m_0, m_1. A random bit $b \xleftarrow{\$} \{0,1\}$ is chosen and $c \leftarrow \mathsf{Enc}_{pk}(m_b)$ is given to $\mathcal{A}$.*
4) Finish. *The adversary outputs $b' \in \{0,1\}$.*

The advantage of the adversary is $|\Pr[b' = b] - \frac{1}{2}|$.

We do not explicitly consider leakage during decryption because we assume that decryption is deterministic (and so any such leakage is captured by leakage queries as allowed above).

III. Random Subspaces are Leakage Resilient

We introduce a linear-algebraic tool that is crucial to our leakage-resilient constructions. Roughly, we show that random subspaces are resilient to continual leakage in an information-theoretic sense. We believe this tool is interesting in its own right, and may find further applications.

For concreteness we work in $\mathbb{Z}_p^m$ (p prime), though what we say generalizes to subspaces of arbitrary vector spaces. Fix an arbitrary leakage function f on the ambient space $\mathbb{Z}_p^m$. Let $\mathbf{X} \in \mathbb{Z}_p^{m \times \ell}$ be a random matrix of rank $\ell \geq 2$; abusing notation, we also let $\mathbf{X}$ denote the linear subspace (of dimension ℓ) generated by the columns of this matrix. We show that when the output length of f is sufficiently short, then the random variables $(\mathbf{X}, f(\mathbf{v}))$ (for random $\mathbf{v} \in \mathbf{X}$) and $(\mathbf{X}, f(\mathbf{u}))$ (for random $\mathbf{u} \in \mathbb{Z}_p^m$) are statistically close. Since $f(\mathbf{u})$ is independent of $\mathbf{X}$ (and hence leaks no information about $\mathbf{X}$), we conclude that when the output length of f is sufficiently short then $f(\mathbf{v})$ for a random $\mathbf{v} \in \mathbf{X}$ leaks almost no information about $\mathbf{X}$.

We also consider the case where the leakage function is applied to *two* random vectors in $\mathbf{X}$. In other words, the leakage function is applied to a random dimension-2 subspace $\mathbf{Y} \subseteq \mathbf{X}$ (in matrix notation $\mathbf{Y} = \mathbf{X} \cdot \mathbf{T}$, where $\mathbf{T} \in \mathbb{Z}_p^{\ell \times 2}$ is a random matrix of rank 2). In this case, we need to assume that the subspace $\mathbf{X}$ has dimension $\ell \geq 4$. More generally, the dimension of $\mathbf{X}$ needs to be at least twice the dimension of the subspace leaked. Namely, if the leakage function is applied to a random subspace of $\mathbf{X}$ of dimension d, then we need $\mathbf{X}$ to have dimension $\ell \geq 2d$.

Theorem 1. *Let p be prime, $\epsilon > 0$, and $m, \ell, d \in \mathbb{N}$ with $2 \leq 2d \leq \ell \leq m$. Fix an arbitrary function $f : \mathbb{Z}_p^{m \times d} \to W$ with $|W| \leq 4 \cdot (1 - 1/p) \cdot p^{\ell - (2d-1)} \cdot \epsilon^2$. Then*

$$\mathsf{dist}\big((\mathbf{X}, f(\mathbf{X} \cdot \mathbf{T})) , (\mathbf{X}, f(\mathbf{Y})) \big) \leq \epsilon \,, \tag{1}$$

where $\mathbf{X} \xleftarrow{\$} \mathbb{Z}_p^{m \times \ell}$, $\mathbf{T} \xleftarrow{\$} \mathrm{Rk}_d(\mathbb{Z}_p^{\ell \times d})$, and $\mathbf{Y} \xleftarrow{\$} \mathbb{Z}_p^{m \times d}$.

Note that for $d = 1$ the leakage function $f(\mathbf{v})$ can leak almost the entire $\mathbf{v}$. This is the case, since according to our parameters, $|\mathbf{v}| = \ell \cdot \log p$, and as long as the leakage size is at most $\log |W| \leq (\ell - 1) \log p + 2 \log \epsilon$, the leakage $f(\mathbf{v})$ hides the subspaces $\mathbf{X}$, up to an ϵ factor. Taking p to be super-polynomial in the security parameter and taking $\epsilon = 1/p$ (which are the parameters used in this paper), we get that as long as the leakage size is at most $(\ell - 3) \log p$ the leakage $f(\mathbf{v})$ statistically hides the subspaces $\mathbf{X}$.

The proof of the above theorem (see [8]) relies on the "crooked leftover hash lemma" of [16], [6].[3] In the full version of this work we also give a simpler and more intuitive proof that Equation (1) holds for $|W| = O\left(q^{\frac{\ell-(2d-1)}{2}} \cdot \epsilon^{3/2}\right)$.

[3]We thank Yevgeniy Dodis, Gil Segev, and Daniel Wichs for pointing out this analysis to us.

IV. A CML-Secure Encryption Scheme

In this section, we present public-key encryption schemes $\mathcal{L}$ and $\mathcal{L}^*$ that are secure against continual leakage. The schemes are parameterized by an integer ℓ that allows for a tradeoff between the sizes of keys and ciphertexts, and the tolerable leakage rate. Scheme $\mathcal{L}$ is secure under the decisional linear assumption in bilinear groups, as long as at most $(1/2 - o(1))$ fraction of the memory leaks between consecutive key updates. The security of $\mathcal{L}^*$ relies on the less standard SXDH assumption (described below), however it can tolerate leakage of a $(1 - o(1))$ fraction of the secret key between key updates. $\mathcal{L}^*$ is also somewhat simpler than $\mathcal{L}$. However, since it is based on a less standard assumption, we will focus our attention mostly on $\mathcal{L}$. We note that the analyses of $\mathcal{L}$ and $\mathcal{L}^*$ are very similar and differ only in the value of the parameter d used when applying Theorem 1 (we use $d = 1$ for $\mathcal{L}^*$, and $d = 2$ for $\mathcal{L}$). Scheme $\mathcal{L}$ is described in Section IV-A, and scheme $\mathcal{L}^*$ is described in Section IV-B. We provide an outline of the security proof for $\mathcal{L}$ in Section IV-C. We refer the reader to the full version [8] for further details.

We show that our schemes are resilient to leakage of constant rate between updates, as well as small leakage (of sub-constant rate) from the key-update and key-generation procedures. This corresponds to $\rho_M = \Omega(1)$ and $\rho_G = \rho_U = o(1)$ in Definition 1.

A. Scheme $\mathcal{L}[\ell]$

Our first scheme is parameterized by an integer ℓ, with keys and ciphertexts having length linear in ℓ. The scheme has leakage rate $\rho_M = \frac{\ell-6-\gamma}{2\ell}$ for all $\gamma > 0$; thus, taking ℓ large enough we can tolerate leakage $(1/2 - \epsilon)$ for any desired ϵ. (Taking ℓ asymptotically increasing gives $\rho_M = (1/2 - o(1))$.)

We can also tolerate leakage during key updates. Specifically, we show that our scheme achieves $\rho_U = O\left(\frac{\log k}{\ell \log p}\right)$, where p is the order of the group in which we work. Since the secret key in our scheme has length $O(\ell \log p)$, this translates to tolerating an absolute leakage of $O(\log k)$ bits during each key update. In fact, our proof can be generalized in a straightforward manner to achieve $\rho_U = O\left(\frac{\log T(k)}{\ell \log p}\right)$ under the assumption that the decisional linear assumption is hard for adversaries that run in time $\mathrm{poly}(T(k))$. It is easy to show that the scheme also tolerates the same amount of leakage (i.e., $O(\log k)$ bits under a standard assumption or $O(\log T(k))$ bits under a strengthened one) from the key-generation procedure.

To put these results in context, we note that tolerating leakage of $O(\log k)$ bits *once* is trivial, since this leakage can be guessed. However, tolerating leakage of $O(\log k)$ bits *repeatedly* is significantly harder.

Before describing the scheme, we introduce some notation. If $g \in \mathbb{G}$ is an element in a group $\mathbb{G}$ of order p and $\mathbf{X} \in \mathbb{Z}_p^{m \times n}$ is a matrix, then $g^{\mathbf{X}} \in \mathbb{G}^{m \times n}$ denotes the matrix where $(g^{\mathbf{X}})_{i,j} = g^{(\mathbf{X})_{i,j}}$. Observe that given a matrix $g^{\mathbf{X}} \in \mathbb{G}^{m \times n}$ and a matrix $\mathbf{A} \in \mathbb{Z}_p^{\ell \times m}$ one can efficiently compute $g^{\mathbf{AX}}$ by working "in the exponent". Note further that if $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map, then given matrices $g^{\mathbf{X}} \in \mathbb{G}^{m \times n}$ and $g^{\mathbf{Y}} \in \mathbb{G}^{n \times \ell}$ one can efficiently compute the matrix $e(g, g)^{\mathbf{XY}} \in \mathbb{G}_T^{m \times \ell}$.

Our construction here relies on the (matrix form of the) decisional linear assumption in bilinear groups [7], [31], which states that the ensembles $\{g^{\mathbf{X}}\}$ and $\{g^{\mathbf{Y}}\}$ are computationally indistinguishable if $\mathbf{X}$ is a random rank-d matrix ($d \geq 2$) over $\mathbb{Z}_p$, and $\mathbf{Y}$ is a random rank-$(d + t)$ matrix ($t \geq 0$) over $\mathbb{Z}_p$ of the same dimensions.

Let us overview the main ideas behind the structure of our scheme. We start by recalling a simplified version of the IBE scheme of [9], and then explain what changes need to be made to achieve leakage resilience. (For now, we do not try to achieve IBE and thus we ignore this aspect of the scheme.)

Let $\mathbb{G}$ be a group of prime order p, with $g \in \mathbb{G}$ a generator. In the scheme of [9], the public key is $g^{\mathbf{A}}$ where $\mathbf{A} \in \mathbb{Z}_p^{2 \times \ell}$ is chosen at random. To encrypt a "0", the sender chooses a random (column) vector $\mathbf{r} \in \mathbb{Z}_p^2$ and computes the ciphertext $g^{\mathbf{r}^T \mathbf{A}}$; to encrypt a "1", the sender chooses a random vector $\mathbf{u} \in \mathbb{Z}_p^{\ell}$ and outputs the ciphertext $g^{\mathbf{u}^T}$. Indistinguishability (based on the decisional linear assumption) follows by looking at the $3 \times \ell$ matrix defined by the public key and ciphertext: when a 0 is encrypted, this matrix takes the form $g^{\mathbf{X}}$ for $\mathbf{X} \in \mathbb{Z}_p^{3 \times \ell}$ of rank 2, but when a 1 is encrypted the matrix takes the form $g^{\mathbf{X}}$ for $\mathbf{X}$ of rank 3 (with overwhelming probability).

The next question, of course, is figuring out how to decrypt such ciphertexts. Notice that any non-zero vector $\mathbf{y}$ in the kernel of $\mathbf{A}$ can be used to decrypt: given a ciphertext $g^{\mathbf{v}^T}$ the receiver can compute $g^{\mathbf{v}^T \cdot \mathbf{y}}$ and check if the result is g^0. This will always be the case for encryptions of 0 and will only happen with negligible probability for encryptions of 1. This suggests that any such $\mathbf{y}$ can be used as a secret key. In [9], due to constraints related to the IBE property, $g^{\mathbf{y}}$ is used as the secret key and decryption is performed using the bilinear map. We will use a similar (but not identical) secret key in our scheme, and the main challenge is then to provide a mechanism for this key to be updated while tolerating leakage.

In order to allow for updates, we use a secret key of the form $g^{\mathbf{Y}}$, where $\mathbf{Y} = [\mathbf{y}_1 | \mathbf{y}_2] \in \mathbb{Z}_p^{\ell \times 2}$, and $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Z}_p^{\ell}$ are random vectors in the kernel of $\mathbf{A}$. Decryption of a ciphertext $g^{\mathbf{v}^T}$ is done by checking whether $e(g, g)^{\mathbf{v}^T \mathbf{Y}}$ is equal to $e(g, g)^{\mathbf{0}}$.

The key-update operation is done by "rotating" the matrix $\mathbf{Y}$; i.e., the receiver samples a random 2×2 full-rank matrix $\mathbf{R} \in \mathbb{Z}_p^{2 \times 2}$ and sets the new secret key to $g^{\mathbf{Y} \cdot \mathbf{R}}$. Intuitively, the linear assumption implies that running the update operation is indistinguishable from sampling a fresh random secret key, which turns out to be a useful property.

The scheme $\mathcal{L}[\ell]$ is presented in Figure 1.

Encryption scheme $\mathcal{L}[\ell]$

Parameters. Take groups $\mathbb{G}, \mathbb{G}_T$ of prime order p, with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$ (and so $e(g,g)$ is a generator of $\mathbb{G}_T$). Let $\ell \geq 7$.

Key generation. The key-generation algorithm samples $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{2\times\ell}$ and $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \ker(\mathbf{A})$, and sets $\mathbf{Y} = [\mathbf{y}_1|\mathbf{y}_2]$. It outputs $pk = g^{\mathbf{A}}$ and $sk = g^{\mathbf{Y}}$.

Key update. To update the secret key $sk = g^{\mathbf{Y}} \in \mathbb{G}^{\ell\times 2}$, sample $\mathbf{R} \xleftarrow{\$} \mathrm{Rk}_2(\mathbb{Z}_p^{2\times 2})$ and then set $sk' = g^{\mathbf{Y}\cdot\mathbf{R}}$.

Encryption. Given a public key $pk = g^{\mathbf{A}} \in \mathbb{G}^{2\times\ell}$, encryption of 0 is done by sampling $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^2$ and outputting the ciphertext $c = g^{\mathbf{r}^T\cdot\mathbf{A}}$. Encryption of 1 is done by choosing $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^\ell$ and outputting $c = g^{\mathbf{u}^T}$.

Decryption. To decrypt a ciphertext $c = g^{\mathbf{v}^T}$ given a secret key $sk = g^{\mathbf{Y}}$, compute $e(g,g)^{\mathbf{v}^T\cdot\mathbf{Y}}$ and output 0 iff the result is $e(g,g)^{\mathbf{0}}$.

Figure 1: A CML-secure encryption scheme based on the decisional linear assumption.

Theorem 2. *Under the decisional linear assumption, for every $\ell \geq 7$ and all $\gamma, c > 0$, encryption scheme $\mathcal{L}[\ell]$ (described in Figure 1) is secure in the continual leakage model with leakage rate*

$$(\rho_G, \rho_U, \rho_M) = \left(\frac{c\cdot\log k}{4\ell\cdot\log p},\ \frac{c\cdot\log k}{(2\ell+4)\cdot\log p},\ \frac{\ell-6-\gamma}{2\ell}\right).$$

An outline of the proof appears in Section IV-C.

B. The Scheme $\mathcal{L}^*[\ell]$

In this section we describe a scheme $\mathcal{L}^*[\ell]$ that is simpler than $\mathcal{L}[\ell]$. While we initially used this scheme for explanatory purposes only, Daniel Wichs pointed out to us that the scheme can in fact be proven secure under the SXDH assumption in bilinear groups. Moreover, the scheme enjoys better leakage resilience: it achieves security for leakage rate $(1 - o(1))$ between key updates, and tolerates the same amount of leakage during key generation and key updates as $\mathcal{L}[\ell]$ does.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p such that there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The SXDH assumption asserts that the decisional Diffie-Hellman (DDH) assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$. Translating into matrix notation, the SXDH assumption implies that it is hard to distinguish random rank-1 matrices from random rank-2 matrices (of the same dimensions) in the exponent, in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

We present the scheme $\mathcal{L}^*[\ell]$ in Figure 2 and state its properties below, without proof.

Theorem 3. *Under the* SXDH *assumption, for every $\ell \geq 3$ and $\gamma, c > 0$, encryption scheme $\mathcal{L}^*[\ell]$ (described in Figure 2) is secure in the continual leakage model with leakage rate*

$$(\rho_G, \rho_U, \rho_M) = \left(\frac{c\cdot\log k}{2\ell\cdot\log p},\ \frac{c\cdot\log k}{(\ell+2)\cdot\log p},\ \frac{\ell-2-\gamma}{\ell}\right).$$

The proof of this theorem is almost identical to that of Theorem 2 (see outline in Section IV-C below). The only difference is in our application of Theorem 1, where we use $d = 2$ in the proof of Theorem 2 and $d = 1$ in the proof of Theorem 3.

Encryption scheme $\mathcal{L}^*[\ell]$

Parameters. Take groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p, with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let g_1, g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively (and so $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$). Let $\ell \geq 3$.

Key generation. The key-generation algorithm samples $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $\mathbf{y} \xleftarrow{\$} \ker(\mathbf{a})$. It then sets $pk = g_1^{\mathbf{a}}$ and $sk = g_2^{\mathbf{y}}$.

Key update. To update the secret key $sk = g_2^{\mathbf{y}} \in \mathbb{G}_2^\ell$, sample $r \xleftarrow{\$} \mathbb{Z}_p$ and set $sk' = g_2^{r\mathbf{y}}$.

Encryption. Given a public key $pk = g_1^{\mathbf{a}} \in \mathbb{G}_1^\ell$, encryption of 0 is done by sampling $r \xleftarrow{\$} \mathbb{Z}_p$ and outputting $c = g_1^{r\mathbf{a}^T}$. Encryption of 1 is done by choosing $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^\ell$ and outputting $c = g_1^{\mathbf{u}^T}$.

Decryption. To decrypt a ciphertext $c = g_1^{\mathbf{v}^T}$ given a secret key $sk = g_2^{\mathbf{y}}$, compute $e(g_1, g_2)^{\mathbf{v}^T\cdot\mathbf{y}}$ and output 0 iff the result is $e(g_1, g_2)^0$.

Figure 2: A CML-secure encryption scheme based on the SXDH assumption.

C. Proof Outline of Theorem 2

As explained above, our scheme is resilient to leakage from memory, from the key-update procedure and from the key generation procedure. We will leave handling leakage from key generation to the end. Thus assume that $\rho_G = 0$, until we mention otherwise.

Let us start by considering a possible proof for standard CPA security of $\mathcal{L}[\ell]$ (as already sketched above). The public key $g^{\mathbf{A}}$ contains a random $2 \times \ell$ matrix (in the exponent), and the challenge ciphertext is either a linear combination of the rows of $\mathbf{A}$ or a random vector of length ℓ. The joint distribution of the public key and ciphertext (we can consider this to be a matrix $\mathbf{V} \in \mathbb{Z}_p^{3\times\ell}$ whose first 2 rows are the matrix $\mathbf{A}$ and its last row is the challenge $\mathbf{v}^T$) is statistically close to either a random rank-2 matrix or a random rank-3 matrix (in the exponent), respectively. An adversary that distinguishes these distributions, therefore, immediately breaks the decisional linear assumption.

In order to make the proof work when the adversary can leak information about the secret key, we need to somehow simulate the leakage from the secret key. This seems self defeating, as having a secret key for the scheme should mean

that the CPA adversary becomes useless (as we can use the secret key and decrypt the ciphertext ourselves).[4] We show that this discouraging intuition is not correct. To this end, we rely on the fact that decryption in our scheme is not perfectly correct, but instead has negligible probability of error.

Consider, at this point, the case where no update queries are made, i.e., the same secret key is used throughout the attack. At a very high level, what we will do is show how to generate keys and a challenge ciphertext in such a way that the secret key, while being appropriately distributed respective to the public key, is useless against our challenge ciphertext. Namely, the distribution (pk, sk) is proper and the distribution (pk, c) is proper, but (pk, sk, c) is far from being proper, in the sense that the secret key sk is useless in decrypting the specific ciphertext c (specifically, c will always decrypt to 0 using sk).[5] Hence, a CPA adversary that decrypts c correctly can actually be useful in breaking the decisional linear assumption.

We then show that if the amount of leakage is sufficiently bounded, an adversary cannot gain sufficient information about the secret key sk to discriminate the ciphertext c from a random ciphertext, and thus should indeed decrypt c correctly (with non-negligible probability). We explain this in detail below and then explain how to extend this to the continual leakage scenario where key updates occur (and thus multiple secret keys are used and are leaked during the attack).

Let us first explain how to generate keys and a challenge ciphertext in such a way that the keys are properly distributed but are still useless against the challenge ciphertext, so that a successful CPA adversary can be used to break the decisional linear assumption. An instance of the decisional linear assumption is a matrix $g^{\mathbf{C}}$, where $\mathbf{C} \in \mathbb{Z}_p^{3\times 3}$ is either rank-2 or rank-3. We will use a successful CPA adversary $\mathcal{A}$ to construct a PPT algorithm $\mathcal{B}$ that distinguishes between the case that $\mathbf{C}$ is of rank 2 and the case that $\mathbf{C}$ is of rank 3.

The algorithm $\mathcal{B}$, on input $g^{\mathbf{C}}$, does the following. It samples $\ell - 3$ random vectors $\{\mathbf{x}_i\} \xleftarrow{\$} \mathbb{Z}_p^\ell$ (with all but negligible probability, these vectors are linearly independent), and generates a $3 \times \ell$ matrix $g^{\mathbf{V}}$ that has the following properties: (1) If $\mathbf{C}$ is a random rank-2 matrix, then $\mathbf{V}$ is a random rank-2 matrix; (2) If $\mathbf{C}$ is a random rank-3 matrix, then $\mathbf{V}$ is a random rank-3 matrix; (3) The vectors $\{\mathbf{x}_i\}$ are uniformly distributed in the kernel of $\mathbf{V}$. This can be done using linear algebra, as follows. Let $\mathbf{X}$ denote the $\ell \times (\ell - 3)$ matrix whose i^{th} column is $\mathbf{x}_i$. Let $\mathbf{B}$ be a basis for the linear subspace $\{\mathbf{w}^T \in \mathbb{Z}_p^\ell : \mathbf{w}^T \cdot \mathbf{X} = \mathbf{0}\}$. Note that $\mathbf{B}$ is efficiently computable given $\mathbf{X}$. Setting $g^{\mathbf{V}} = g^{\mathbf{C}\cdot\mathbf{B}}$, it holds that $\mathbf{V}$ has the same rank as $\mathbf{C}$. By symmetry, the vectors $\{\mathbf{x}_i\}$ are uniformly distributed in $\ker(\mathbf{V})$ as required.

The algorithm $\mathcal{B}$ feeds the adversary $\mathcal{A}$ with the first two rows of $g^{\mathbf{V}}$ (denoting it by $g^{\mathbf{A}}$) as the public key. It generates a secret key $g^{\mathbf{y}_1}, g^{\mathbf{y}_2}$ by choosing at random $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \{\mathbf{X} \cdot \mathbf{t} : \mathbf{t} \in \mathbb{Z}_p^{\ell-3}\}$, i.e., randomly sampling from the column-span of $\mathbf{X}$. Then, it feeds $\mathcal{A}$ with the challenge ciphertext $g^{\mathbf{v}^T}$, which is the last row of $g^{\mathbf{V}}$. Notice that our secret key will always decrypt the challenge ciphertext $g^{\mathbf{v}^T}$ to 0, even if $\mathbf{v}^T$ is independent of the rows of $\mathbf{A}$ and hence should be decrypted to 1 (thus, our secret key is "crippled" w.r.t. the ciphertext $g^{\mathbf{v}^T}$). A CPA adversary that succeeds in decrypting c correctly (in spite of getting leakage from a "crippled" secret key) will, therefore, break the decisional linear assumption by determining if $\mathbf{V}$ (and hence $\mathbf{C}$) is rank-2 or rank-3.

We next must explain why the adversary $\mathcal{A}$, which is given leakage from our "crippled" secret key, should nevertheless decrypt c correctly (with noticeable probability). To this end, we use Theorem 1 to show that a small enough leakage *statistically* hides the subspace $\mathbf{X}$ that the secret key is sampled from, making it statistically hard to distinguish this distribution of the secret key and the legal distribution where $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \ker(\mathbf{A})$. Thus, the adversary cannot use the leakage to distinguish between our "crippled" secret key and a genuine one. This completes the proof for the case of non-continual leakage.

We now address the fact that the leakage is continual, namely, that there is a sequence of phases in which leakage occurs, each followed by a refresh operation. At this point, however, we still assume that the key-update procedure does not leak (i.e., $\rho_U = 0$).

While our "legal" key-update procedure never changes the column-span of the secret key (recall that the update procedure takes $sk = g^{\mathbf{Y}}$ and outputs $sk' = g^{\mathbf{Y}\cdot\mathbf{R}}$ for an invertible $\mathbf{R}$), we notice that this is computationally indistinguishable from an update procedure that re-samples new vectors $\mathbf{y}_1, \mathbf{y}_2$ from the column-span of $\mathbf{X}$ (i.e., an update procedure that sets $sk' = g^{\mathbf{X}\cdot\mathbf{T}}$, for a properly sampled matrix $\mathbf{T}$, regardless of the previous sk). Indistinguishability holds as the legal distribution on keys can be described by setting $sk' = g^{\mathbf{X}'\cdot\mathbf{T}}$, where $\mathbf{X}'$ is a random rank-2 matrix; thus, distinguishing $g^{\mathbf{X}}$ and $g^{\mathbf{X}'}$ is hard by the decisional linear assumption, and this holds even in the presence of the matrix $\mathbf{A}$. (Jumping ahead, it is this indistinguishability argument that becomes troublesome when leakage from the key-update procedure is allowed.) Note that this "improper" update can be simulated, since we explicitly know $\mathbf{X}$. This enables us to apply the above argument consecutively: at each phase, we will leak from new vectors in the column-span of $\mathbf{X}$. Applying Theorem 1, the view of the adversary is statistically close to the case where the leakage function is applied to $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \ker(\mathbf{A})$ which, in turn, is indistinguishable from the distribution of "legal" keys.

We are left with treating leakage from the update process (i.e., $\rho_U > 0$). Any amount of such leakage seems to completely break the indistinguishability argument of the previous paragraph. Intuitively, the adversary is getting leakage from the input to the update procedure (including the random tape

[4] A similar problem arises in most works on leakage resilience, and is addressed in various manners.

[5] It is here that we use the fact that our scheme is not perfectly correct.

used), and thus may have some information on what the legal output should be. Once the adversary sees some memory leakage from an improper secret key, used for the security reduction, it can compare it to what it knows about the legal secret key, thus catching any attempt for a switch of distributions as above. In order to overcome this barrier, we must come up with a way to simulate the leakage from the update, in such a way that makes $\mathcal{A}$ behave similarly in spite of the discrepancy between the distributions.

The key observation is that whether or not $\mathcal{A}$ "behaves similarly" is an efficiently checkable event. We can simulate the remainder of the security game, using the proposed improper secret key and some candidate leakage value (as if everything from this point on is done legitimately) and see if $\mathcal{A}$'s success probability decreases or not. If the number of bits being leaked is small enough, specifically $O(\log k)$, we can efficiently go over all possible values until we find one that works.

There is one small problem: what if *no* leakage value works? In this case, we recall again that the improper distribution on keys used in our reduction is indistinguishable from the original one. Therefore, since in the original distribution there is always a good leakage value — the legal value[6] — this should also be the case with the improper one. This holds since the event of not finding an acceptable leakage value is also efficiently checkable (again, by simulating $\mathcal{A}$). A change in behavior in this respect yields a distinguisher between the real and improper distributions, which is impossible under our hardness assumption.

We obtain, therefore, that leakage of $O(\log k)$ bits from the update procedure can indeed be tolerated. This can be generalized in a straightforward manner to imply a tolerable leakage of $O(\log T(k))$ bits, if we assume that the decisional linear assumption is hard for (roughly) $T(k)$-time adversaries.

The last thing to do is handle leakage from the key-generation step. At this point, however, it is clear how this can be done (for $O(\log k)$ bits of leakage). We use the same technique we use for updates: after generating our initial secret key, we go over all possible values of key-generation leakage, and find one for which $\mathcal{A}$ works well.

V. A CML-Secure IBE Scheme

We present an identity based encryption (IBE) scheme that is secure against continual leakage. In an IBE scheme, any string (or *identity*) can be used as a public key (combined with a set of *public parameters* that are known to all). A special *master secret key* can be used to generate specific secret keys per identity. Our IBE scheme is very similar to our scheme $\mathcal{L}$ which, as we mentioned above, originated from the IBE scheme of [9].

[6]In fact, this is only true with noticeable probability over the sampling of the secret key, and therefore we may need to try a few keys before we find a good one.

We recall that in the context of IBE, there are two types of "entities" holding secrets: a trusted authority holding the master secret key that enables producing specific secret keys per user; and individual users, each holding their own secret key. In this work, we only allow leakage from the memory of the individual users. For a more detailed discussion of identity-based encryption schemes and associated security notions under continual leakage, we refer the reader to the full version [8].

In our construction, we consider an IBE scheme where the set of identities is $\{0,1\}^m$. The public parameters of the scheme include a set of $2m+1$ matrices $(g^{\mathbf{A}_0},\ \{g^{\mathbf{A}_{i,b}}\}_{i\in[m],b\in\{0,1\}})$ of dimension 2×2. Identity $id \in \{0,1\}^m$ is associated with the matrix $\mathbf{A}_{id} = [\mathbf{A}_0\|\mathbf{A}_{id_1}\|\cdots\|\mathbf{A}_{id_m}]$. Specifically, we use $g^{\mathbf{A}_{id}}$ in order to encrypt messages for id. To decrypt, the user corresponding to id uses the secret key $sk_{id}{:=}g^{[\mathbf{y}_1\|\mathbf{y}_2]}$, where $\mathbf{y}_1,\mathbf{y}_2 \xleftarrow{s} \ker(\mathbf{A}_{id})$. In other words, each user is associated with a pair of keys corresponding to $\mathcal{L}[\ell]$ (see Section IV), defined by its identity and the public parameters. We notice that the trusted authority only needs to know $msk = \mathbf{A}_0$ in order to produce secret keys for all users. The encryption and decryption algorithms are similar to those of $\mathcal{L}[\ell]$.

The proof of security is by reduction to the security of $\mathcal{L}[\ell]$ for $\ell = 2m+2$. Specifically we prove that our scheme is *selectively secure* with continual leakage. In selective security [11] we assume that the attacker decides on the identity it wants to attack before seeing the public parameters. Let id^* be that identity. This enables generating the public parameters such that all matrices $\mathbf{A}_{i,1-id^*_i}$ (i.e., all matrices that do not affect $\mathbf{A}_{id^*}$) are explicitly known, and a public key for $\mathcal{L}[2m+2]$ is "embedded" in the public parameters as the matrix $g^{\mathbf{A}_{id^*}}$. This enables the adversary to know everything about all other identities (since it is sufficient to explicitly know one sub-matrix of $\mathbf{A}_{id}$ in order to generate a corresponding secret key) while knowing nothing about id^*. Thus, an adversary that breaks the security of id^*, even with continual leakage, actually breaks the security of $\mathcal{L}[2m+2]$.

The leakage guarantees of our scheme are formally stated in the theorem below. We note that since we cannot allow any leakage from the master secret key, the relevant leakage parameters are leakage during the update and memory leakage of the individual users, denoted ρ_U, ρ_M, respectively.

Theorem 4. *Under the decisional linear assumption, for every polynomially bounded m, the* IBE *scheme described above is selective secure under chosen plaintext attack, in the* CML *model, with the following parameters. For all $\gamma, c > 0$ the scheme is secure with leakage rate*

$$(\rho_U, \rho_M) = \left(\frac{c\cdot\log k}{\ell\cdot\log p}, \frac{m-2-\gamma}{2(m+1)}\right) .$$

VI. A CML-Secure Signature Scheme

We show how to use any encryption scheme secure with continual leakage (such as the encryption scheme $\mathcal{L}[\ell]$ presented in Section IV) to construct a signature scheme that is secure with continual leakage.

When discussing continual leakage in the context of signature schemes, we must also take into account the signing operation. This operation involves the secret key and additional secret randomness and may potentially leak extra information. Therefore, we are concerned with an additional parameter of leakage resilience, namely the leakage rate during the signing process, which we denote by ρ_S. This is defined as the total amount of information leaked during a signing process, divided by the total size of the secret key and secret randomness used by the signing procedure. For a more detailed discussion and for the full constructions, see the full version [8].

A. No Leakage During the Signing Process

Our first construction applies the techniques of [25] to our encryption scheme to obtain a signature scheme that preserves the leakage guarantees but does not allow any leakage during the signing process (i.e., $\rho_S = 0$). The signing key contains a secret key sk for the encryption scheme, and the verification key contains the public key pk as well as an additional public key pk' for a (not necessarily leakage-resilient) public-key encryption scheme, and a common random string for a non-interactive simulation-sound zero-knowledge (NIZK) proof system. A signature on a message m contains an encryption of sk using pk', and a simulation-sound NIZK proof that the contents of the ciphertext are indeed a valid secret key for the leakage-resilient encryption scheme. The dependence on the message m comes from properly defining the language for the simulation-sound NIZK proof, as done in [25].

We show that we can use a successful forger for this scheme to break the security of the leakage-resilient encryption scheme. In the security reduction we follow the outline of the proof in [25]: We simulate all NIZK proofs and sample pk' together with a respective secret key sk'. We can then provide valid-looking signatures for a successful forger without it being able to tell the difference. Then we can take its successful forged signature, which must contain an encryption of a valid secret key for the leakage-resilient encryption scheme, decrypt it using sk', and obtain a valid sk that enables breaking the security of the leakage-resilient encryption scheme. A statement of the result follows.

Theorem 5 (Informal)**.** *Consider a semantically secure public-key encryption scheme in the* CML *model, with leakage rate* (ρ_G, ρ_U, ρ_M)*. Then, under standard cryptographic assumptions, there exists a signature scheme that is existentially unforgeable under adaptive chosen message attacks in the* CML *model, with leakage rate* $(\rho_G, \rho_U, \rho_S, \rho_M)$*, where* $\rho_S = 0$*.*

B. Tolerating Leakage During the Signing Process

Our second construction tolerates leakage from the signing algorithm. Recall that in our previous construction each signature consists of an encryption of the secret key, together with a simulation sound NIZK proof. Note that if the randomness used for the encryption or the NIZK is partially leaked, then the secret key can be leaked entirely.

Thus, we change the signing algorithm as follows. Instead of using a (simulation-sound) NIZK proof system, we use a (non-interactive) argument system with short proofs; i.e., proofs that are significantly shorter than the witness size. Intuitively, even though we cannot say much about what information might be leaked by such a proof, we *can* bound the leakage by the length of the proof. We note that all known short non-interactive argument systems [26], [29], [5] are in the random oracle model. However, we do not use random oracles explicitly anywhere in our proof.

In addition, instead of using a standard encryption scheme, we use a special "dual-mode encryption scheme", that (depending on the public key) may lose most information about the plaintext (information theoretically). More specifically, we "encrypt" the secret key by using a family of lossy trapdoor functions [32], in a particular way. In the security proof, we set the public key so that the lossy functions corresponding to the forged message m are all injective, which results with the encryption being invertible; whereas the lossy functions corresponding to any other message are not all injective, which results with a "lossy encryption". This type of analysis achieves only a weaker notion of security, called a-priori-message unforgeability, where the adversary is given a random "challenge" message m in advance, can request signatures on any messages other than m, and then succeeds only if it outputs a forgery on m. We then apply a recent transformation from [9] to obtain a scheme satisfying existential unforgeability. We refer the reader to the full version [8] for more details and for a formal statement of our results (which requires additional terminology).

We note that, interestingly, in order to allow leakage from the signing algorithm, we make our scheme leak *more*, since in our new scheme, the signatures themselves leak information about the signing key. Thus the scheme does not even conform with the standard definition of security (without leakage). In the CML model, however, this caveat is tolerable, since we have a method for refreshing the key. We have to require, however, that the signing-key is periodically refreshed, even regardless of any adversarial action, resulting in a variant of CML security.

Acknowledgements

We thank Yevgeniy Dodis, Gil Segev, and Daniel Wichs who (independently) pointed us to the crooked leftover hash lemma for an alternative (and better) analysis in the proof of Theorem 1. We thank Daniel Wichs for pointing out that our

proof of security goes through with better parameters if we rely on the SXDH assumption. Thanks, guys!

We thank Shafi Goldwasser for various fruitful discussions, comments, and advice; and Adam Tauman Kalai for helpful comments. Adam, thanks for your positive contribution (in absolute value, at least).

Work of the first and third authors was done in part while being hosted by Microsoft Research (New England). Work of the third author was supported in part by NSF grant #0627306. Work of the fourth author was done while at IBM.

REFERENCES

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, 2009.

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *Advances in Cryptology — Eurocrypt 2010*, volume 6110 of *LNCS*, pages 113–134. Springer, 2010.

[3] J. Alwen, Y. Dodis, and D. Wichs. Public key cryptography in the bounded retrieval model and security against side-channel attacks. In *Advances in Cryptology — Crypto 2009*, volume 5677 of *LNCS*, pages 1–17. Springer, 2009.

[4] R. Anderson. Two remarks on public-key cryptology. Invited lecture, ACM CCCS 1997. Available at http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf.

[5] B. Barak and O. Goldreich. Universal arguments and their applications. *SIAM J. Computing*, 38(5):1661–1694, 2008.

[6] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 335–359. Springer, 2008.

[7] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[8] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. Cryptology ePrint Archive, Report 2010/278, 2010. Full version of this paper.

[9] Z. Brakerski and Y. Tauman Kalai. A framework for efficient signatures, ring signatures, and identity-based encryption in the standard model. Cryptology ePrint Archive, Report 2010/086, 2010.

[10] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. In *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of *LNCS*, pages 453–469. Springer, 2000.

[11] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, 2007.

[12] Y. Dodis, S. Goldwasser, Y. Tauman Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *7th Theory of Cryptography Conference — TCC 2010*, volume 5978 of *LNCS*, pages 361–381. Springer, 2010.

[13] Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs. Cryptography against continuous memory attacks. These proceedings.

[14] Y. Dodis, Y. Kalai, and S. Lovett. On cryptography with auxiliary input. In *41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 621–630. ACM Press, 2009.

[15] Y. Dodis, S. J. Ong, M. Prabhakaran, and A. Sahai. On the (im)possibility of cryptography with imperfect randomness. In *45th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 196–205. IEEE, 2004.

[16] Y. Dodis and A. Smith. Correcting errors without leaking partial information. In *37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 654–663. ACM Press, 2005.

[17] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 293–302. IEEE, 2008.

[18] S. Faust, E. Kiltz, K. Pietrzak, and G. Rothblum. Leakage-resilient signatures. In *7th Theory of Cryptography Conference — TCC 2010*, volume 5978 of *LNCS*, pages 343–360. Springer, 2010.

[19] S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting circuits from leakage: The computationally-bounded and noisy cases. In *Advances in Cryptology — Eurocrypt 2010*, volume 6110 of *LNCS*, pages 135–156. Springer, 2010.

[20] S. Goldwasser and G. Rothblum. How to play mental solitaire under continuous side-channels: A completeness theorem using secure hardware. Crypto 2010. To appear.

[21] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.

[22] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner. Private circuits II: Keeping secrets in tamperable circuits. In *Advances in Cryptology — Eurocrypt 2006*, volume 4004 of *LNCS*, pages 308–327. Springer, 2006.

[23] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology — Crypto 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.

[24] A. Juma and Y. Vahlis. Leakage-resilient key proxies. Crypto 2010. To appear.

[25] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *Advances in Cryptology — Asiacrypt 2009*, volume 5912 of *LNCS*, pages 703–720. Springer, 2009.

[26] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 723–732. ACM Press, 1992.

[27] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology — Crypto '96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[28] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology — Crypto '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[29] S. Micali. Computationally sound proofs. *SIAM J. Computing*, 30(4):1253–1298, 2000.

[30] S. Micali and L. Reyzin. Physically observable cryptography. In *1st Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer, 2004.

[31] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology — Crypto 2009*, volume 5677 of *LNCS*, pages 18–35. Springer, 2009.

[32] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.

[33] 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 2008: 3rd ACM Symp. on Information, Computer, and Communications Security*, pages 56–65. ACM Press, 2008.

[34] K. Pietrzak. A leakage-resilient mode of operation. In *Advances in Cryptology — Eurocrypt 2009*, volume 5479 of *LNCS*, pages 462–482. Springer, 2009.

[35] R. L. Rivest. All-or-nothing encryption and the package transform. In *Fast Software Encryption – FSE '97*, volume 1267 of *LNCS*, pages 210–218. Springer, 1997.