Multi-Location Leakage Resilient Cryptography

Ali Juma[1⋆], Yevgeniy Vahlis[2], and Moti Yung[3]

[1] Mozilla Corporation ajuma@mozilla.com
[2] AT&T Security Research Center evahlis@att.com
[3] Google and Columbia University my123@columbia.edu

Abstract. Understanding and modeling leakage in the context of cryptographic systems (connecting physical protection of keys and cryptographic operation) is an emerging area with many missing issues and hard to understand aspects. In this work we initiate the study of leakage out of cryptographic devices when the operation is inherently replicated in *multiple locations*. This setting (allowing the adversary access to leakage at different locations) arises naturally in cases like protocols, where different parties activate the same cryptographic function, or in the case of a global service providers (like cloud operators) which need to replicate the cryptographic function to allow for accessible and responsive services. We specifically deal with the theoretical setting of "leakage resilient cryptography," (modeling leakage as a bound associated with algorithmic steps), and in the most general model of continual leakage on memory, randomness (and thus computation) with periods of operation and refresh of private keys between them.

We first investigate public-key cryptography, and construct a multi-location leakage resilient signature scheme (with unbounded number of locations) with optimal (i.e., total $n(1 - o(1))$ leakage) in a period, and $O(\log n)$ leakage during updates (n is the key size). The new crucial issue behind our scheme is how to maintain leakage at each location at the level of key leakage in the single location variant, even under parallel adaptive leakage at the different locations. We then construct a shared-symmetric-key authenticated session protocol that is resilient to leakage on both the sender and the receiver, and tolerates $O(\log n)$ bits of leakage per computation. We construct and utilize a single-location pseudorandom generator which is the first to tolerate continual leakage with only an efficient pseudorandom function as a primitive component. This protocol highlights the importance of protocol level "per message synchronization" against leakage adversaries. Interestingly, the construction is secure in spite of the entire randomness used in the refresh processes being publicly available.

1 Introduction

When a cryptographic function/ service is performed at more than one location, and an adversary attacks it, if the adversary has only black-box access to it

⋆ This work was done while at the University of Toronto

and the scheme is stateless or state-synchronized (for correctness), then from security point of view, it does not seem to matter (i.e., and it does not require a new model) whether the scheme is operated from a single location or multiple ones (since the black box information revealed in a sequence of cryptographic application is insensitive to the location). However, when leakage is allowed to be part of the outputs, the adversary gets this added side-channel information [23] and as a result, may have different power, depending on whether the leakage is at a single location or if it comes from multiple locations.

The above observation is the motivation to this work, since the sensitivity to multiple location leakage is important to various systems settings, and we investigate this issue from the "leakage resistance cryptography" perspective. Note that multi location is natural when two parties in a protocol operate the same cryptographic service or when the same function is inserted in various devices (e.g., different cloud servers, different mobile devices within the same organization), etc.

The theme of this work is the design of secure cryptosystems under the existence of multiple locations. We consider both *public key systems*: in particular a signature scheme (while the methods we design may apply in more generality to encryption, etc.), and *symmetric key systems*: in particular session authentication protocols providing sender continual authentication to the receiver, based on shared pseudorandomness. We consider the model of continual leakage with no relaxation, i.e., where the leakage function is not only a result of computation but can be a function of the memory (state) and the randomness (i.e., the computation) as well. In these models, the parties need to go through periods of operation where leakage is given to the adversary and once the accumulated amount of leakage is large enough, the private keys are refreshed between periods (obviously if refresh is not possible, continued leakage may reveal over time the entire key bits, say one by one). We note that the above model is the strongest, compared with more limited types of leakage models that have been considered as well in the literature (such as: leakage in the presence of leakage-free components, leakage where "only computations leak," and only memory leakage (without leaking the randomness used in the cryptographic computation)).

1.1 Multi-Location Leakage Resilient Signature

Since we consider continual leakage we have to make sure that we have a scheme where the total leakage in a period is only a fraction of the state (if the state is l bits long we can allow at most $l(1 - o(1))$ bits of information about the secret to be given to the adversary. Indeed, a few recent schemes have achieved such leakage in a period, allowing logarithmic leakage in the refresh process (e.g. [8, 5]). In this version we concentrate on the signature scheme of [28].

In a multi-location setting, the scheme is replicated in various locations to allow better accessibility o the signing service, say, and necessarily these locations contain related key information (since the verification key is identical regardless of location). If the adversary collects enough information at different locations, each of which by itself is too small to break the key, the cumulative effect may

nevertheless be that the adversary is in possession of enough bits to break the key (e.g., may collect all bits of a single replicated key). Thus, in the naive solution (which allows very limited leakage per location), we may restrict the amount of leakage at each location to be smaller than the total allowed per single location divided by the number of locations. This approach as we will show works sometimes, but sometimes fails!

We then turn into the challenging problem of constructing direct multi-location leakage resilient scheme which allows large leakges per location. Our starting point is a recent signature secure against continual leakage, where the crux behind this scheme is the fact that a key within a period is hidden within a large set of keys and the leakage within a period is simulatable by leakage correlated with a random value rather than the key [28]. Then, the key is refreshed to another value within a large set of keys. A crucial point behind extending the signature to a multi location scheme which is refreshed at all locations periodically (when a bound on signatures at any single location is reached) is extension of the space from which private keys are drawn, so that multiple location leakage will also be simulatable by random values taking the two dimensions of variety of keys, namely, "periods" and "locations," into account when building the space of keys.

1.2 Multi-Location Symmetric-Key Authentication

We next design a session authentication protocol from symmetric key whose goal is to continuously authenticate the sender to the receiver. A natural way of doing it is to base this on a *stream cipher* (i.e., a pseudorandom generator) which is run by both parties. Dziembowski and Pietrzak [12] and Pietrzak [31] gave leakage-resilient stream ciphers in the *only-computation-leaks* model (where the adversary gets a bounded size (logarithmic, in fact) leakage bits each time). Their seminal constructions use two pieces of memory connected by a public channel, and computation alternates between the two pieces. For an authenticated session we have two parties, a sender A and a receiver B, where A is sending message pieces to B, and we wish to ensure that an adversary cannot modify or reorder messages pieces, or insert message pieces of his own without this being detected by B. The adversary obtains leakage from both parties. The existing security definitions of leakage-resilient stream ciphers do not deal with this case at all. In fact, in the existing ciphers, an adversary that can cause parties A and B to "get out of sync" can attack the system and eventually learn the cipher's entire state. This suggests that we need a way to somewhat synchronize the stream cipher computations performed by the two parties.

Our construction, in turn, builds on Pietrzak's stream cipher construction [31], and uses a *single* piece of memory along with a source of strings that are chosen according to distribution of high min-entropy but are not kept secret (*public min-entropy* source), and are rather communicated between the parties. Our stream cipher uses a pseudo-random function generator $F_s : \{0,1\}^n \rightarrow \{0,1\}^{2n}$. The initial secret state is randomly chosen $K_0 \in \{0,1\}^n$. For each $i > 0$, the i-th output is produced and the state is updated. A authenticates the

message using the output, while B generates the next round high entropy string and sends it back for synchronization and update to A. Of course, the adversary controls the public channel and may insert strings of his choice (purporting to be sent by the other party) to induce a party to continue its computation; we show that such tampering by the adversary will be detected by B when it attempts to verify the authenticity of the message pieces he receives; the construction allows continual leakage of logarithmic bits per round.

Remark: We note that Dodis *et al* [8] present a signature scheme which is multi-location leakage resilient (Theorem 7.6 in [8]), although they do not explicitly call it such. However, their scheme does not allow the adversary to obtain leakage on the randomness of signing. In contrast, in this work we define and present a signature scheme with multi-location resilience to full leakage (including signing randomness), and provide several generic theorems for obtaining multi-location leakage bounds for schemes that were intended to support only a single location.

1.3 Related Work

Side channel attacks [23] have often been shown to have devastating effects on the security of cryptographic schemes (some recent attacks that specifically pertain to general memory leakage are described in e.g., [17, 32, 36], and others). As a result a significant effort has been wielded to design cryptographic schemes that provably withstand large classes of such attacks.

The influential theoretical works of Ishai, Sahai, and Wagner [19] and Micali and Reyzin [30] enable us to construct schemes under the "any computation, and only computation, leak information," model, which has led to many recent achievements. In contrast, *memory leakage* [1] (which, in some sense, can be traced to the original works of Shamir [34] and Rivest [33]) are produced as a function of the memory state itself. This type of leakage is orthogonal to computational leakage: an adversary can get memory leakage by probing memories even if the memories are not currently used in any computation (e.g., the cold-boot attacks [17]). For example, the scheme of [9, 8] is secure against memory attacks (even continual), but assumes that the signing process leaks no information. The most general model allows *full leakage* which includes leakage both from processing and memory.

The most demanding case for designing digital signature schemes seems to be the case of adaptive and continual full leakage that is available to the adversary from both computational and memory sources (without protection of sub-steps of computations). However, till recently there are no known schemes which achieve a digital signature scheme in this adversarial setting in the standard model. and without further relaxations. All known schemes with full (memory and processing) leakage either did not have a key update algorithm and thus are not continual (cf., [22]), have a key update algorithm but require some restrictions (e.g., [3, 2] which requires an additional leakage-free master key), or are based on the random oracle model (with a relaxation of the definition of a "time period") [5]. Faust *et al* [13] construct signature schemes resilient to continual leakage in

the only computation leaks model. Recently, two schemes have appeared with continual full leakage [28, 4].

In the private key setting, Dziembowski and Pietrzak [12], and Pietrzak [31] describe the first stream ciphers resilient to continual leakage in the only computation leaks model. Our private key construction uses the works of [12, 31] as a starting point. Many other schemes dealt with the case of designing pseudorandom generation [12, 31, 39, 11].

Faust *et al* [14] give a general compiler using secure hardware that protects an arbitrary circuit against continual leakage that can be modeled as a shallow (AC^0) boolean circuit. Juma and Vahlis [20], and separately Goldwasser and Rothblum [16], give compilers that protect any algorithm against continual leakage (without complexity restrictions), using secure hardware. Recently, Dodis and Pietrzak [11] show how to build continual leakage resilient pseudorandom functions that are secure against non-adaptive leakage. Finally, Lewko *et al* [26, 25] show how to achieve leakage resilient Identity Based Encryption (IBE) and super-logarithmic leakage on key updates in signatures, and Chow *et al* [7] show an efficient leakage resilient IBE. A separate line of work studies strong leakage resilience in information theoretic implementation settings [35].

Finally, we mention a parallel rich line of work on tamper resistant cryptography [6, 27, 24, 29, 18, 15]. Here, an adversary has the ability to modify, rather than observe, the state of the cryptographic primitive. Tamper resistant schemes provide security guarantees, even when the secret state is transformed through a tampering function adversarially chosen from a large class of functions. We remark that the techniques seem to be quite different from the ones employed to achieve leakage resilience. Indeed, studying the relation between tamper resistance and leakage resistance is an important direction.

Roadmap. In Section 2 we present our definitions of multi-location continuous leakage resilient signatures, and constructions. In Section 3 we present a shared key authenticated session protocol that is resilient to continuous leakage, and in Section 4 we present our variant of the Dziembowsky-Pietrzak leakage resilient stream cipher [12].

Notation. We write PPT to denote Probabilistic Polynomial Time. When we wish to fix the random bits of a PPT algorithm M to a particular value, we write $M(x;r)$ to denote running M on input x and randomness r. We write $time_n(M)$ to denote the running time of algorithm M on security parameter n. We use $x \in_{\mathrm{R}} S$ to denote the fact that x is sampled according to a distribution S. Similarly, when describing an algorithm we may write $x \leftarrow_{\mathrm{R}} S$ to denote the action of sampling an element from S and storing it in a variable x. For a randomized algorithm M, we denote by $\mathsf{Rnd}[M]$ the space of its random coins. Namely, if M uses at most n_M random bits in any execution, then $\mathsf{Rnd}[M] = \{0,1\}^{n_M}$.

2 Multi-Location Leakage Resilience in the Public Key Setting

In a multi-location setting, an adversary (or perhaps multiple colluding adversaries) simultaneously mount side channel attacks on multiple devices. When the devices contain unrelated data, and perform unrelated computations, such an attack should be viewed as a single side channel attack on each of the devices, since none of the other devices can provide any information that would help the attacker. The problem becomes much more serious when the devices contain correlated secret data. As an extreme case, consider a situation where multiple identical copies of a secret key are stored on several servers. Even if the cryptographic scheme where the key is used remained secure when the key is partially leaked, an adversary in a multi-location setting may be able to reconstruct the entire key from partial leakage from each of the locations. In this scenario, surely all reasonable security properties of the scheme can be broken. One possible approach to dealing with this apparent limitation is to restrict the *total* amount of leakage that the adversary can obtain across *all* copies of the key. Indeed, for some primitives (such as encryption) we show a straightforward reduction from multi-location leakage resilience with a bound on the total amount of leakage to single location leakage resilience. We note that, perhaps surprisingly, the same straightforward reduction fails for other primitives, such as signature schemes.

Note however that ideally, we want to obtain transformations where the leakage per location remains as large as the leakage bound in the single location setting. In Section 2.1 we give constructions of signature and encryption schemes where the leakage bound per location does not decrease with the number of different locations that maintain an equivalent copy of the key.

Overall, we note that extending multi-location leakage resilience to the continuous setting introduces several subtle challenges that do not appear when considering leakage from only a single location. Before describing these issues, we describe (informally) a generic transformation of a continuous leakage resilient signature scheme into a multi-location variant of itself. This will serve two purposes: firstly, it will help us illustrate the definitional issues that arise in multi-location continual leakage resilience. Secondly, our actual constructions and transformations can all be viewed as variants of the general approach that we describe here.

Consider a signature scheme that is resilient to continual leakage in the single location setting. As we have already discussed, such a scheme must have a key refresh procedure (otherwise, an adversary can eventually obtain the entire key). Moreover, suppose that the refresh procedure produces a new signing key chosen uniformly from the set of all valid keys that correspond to the public verification key that is generated once at the beginning. Now consider the following initialization procedure for an n-location signature scheme:

1. A public-private key pair (vk, sk) is generated using the key generation procedure of the single location scheme.

2. The key refresh procedure is used to produce n random signing keys $sk_1, \ldots, sk_n$, all corresponding to the verification key vk.

3. Location i receives key sk_i. Whenever location i receives a request to update the key, it runs the refresh algorithm on its own key sk_i.

Essentially, each location maintains an independently chosen random signing key for vk, and when the leakage bound is reached for that specific location, the key is randomized. When $n = 1$, this is exactly what happens in the single location setting (and thus for $n = 1$ the security of the scheme trivially follows from its single location security). Consider now what happens when $n > 1$: at first glance it may seem that, because the keys at the different locations are independently chosen, leakage from one location would be completely useless in attacking another location. This turns out to be false for multiple reasons.

2.1 Signature Schemes

A *signature scheme with key update* $\mathcal{SGN}$ consists of four algorithms Kg, Sig, Ver, and Update. The inputs and outputs of Kg, Sig, and Ver are the same as in standard signature schemes. Update takes as input a secret key and a public key and outputs a new element of the secret key space. $\mathcal{SGN} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Update})$ has to satisfy the following property:

- (**Correctness**) For any integers $n, m, i \geq 0$ and any message M, if we compute $(pk, sk_1^{(0)}, \ldots, sk_m^{(0)}) \leftarrow \mathsf{Gen}(1^\kappa, m)$, $sk_0 \leftarrow sk_i^{(0)}$, $sk_1 \leftarrow \mathsf{Update}_{pk}(sk_0)$, …, $sk_n \leftarrow \mathsf{Update}_{pk}(sk_{n-1})$, and $\sigma \leftarrow \mathsf{Sig}(sk_n, M)$, $\mathsf{Ver}(pk, M, \sigma) = 1$ always holds.

We now define multi-location leakage resilience for signatures. Intuitively, the definition is a natural extension of the definitions of leakage resilient signatures that appeared in [28, 5, 8, 4, 10]. Intuitively, the adversary can submit signature queries and leakage queries that are directed at a specific location. For example, the adversary may submit a query that is interpreted as "Have the ith signer sign message m, and obtain side-channel information $f(sk_i, r)$, where r is the randomness used during signing, along with the resulting signature". The other types of queries are location specific signature queries without leakage (to allow longer periods between updates, as discussed in the introduction), and update queries. For update queries, we distinguish between synchronized updates where all locations refresh their keys simultaneously and unsynchronized updates where the adversary instructs the signer at some location i to refresh his key. Finally, the adversary's goal is to produce a valid signature of a message that he has not submitted for signing in any of his queries.

Experiment $\mathsf{ExpMLSIG}(1^n, \mathcal{A}, \mathcal{SIG})$*:*

Setup The adversary submits an integer m, and the challenger runs $\mathsf{Gen}(1^n, m)$ to obtain a public verification key pk, and m location secret keys $sk_1, \ldots, sk_m$.

Queries $\mathcal{A}$ submits queries of the following three types:

- Update queries.

  Unsynchronized setting. Update queries of the form (update, f, i) where f is a circuit satisfying $|f(sk_i, R)| \leq \rho_U(|sk_i| + |R|)$ for any R. If $L_i + |f(sk_i, R)| \leq \rho_M |sk_i|$ holds, the challenger chooses $R \xleftarrow{\$} \mathsf{Rnd}[\mathsf{Update}]$ randomly, computes $sk_i \leftarrow \mathsf{Update}_{pk}(sk_i, R)$, sends $f(sk_i, R)$ back to $\mathcal{A}$, and sets $L_i \leftarrow |f(sk_i, R)|$. Otherwise, the challenger aborts.

  Synchronized setting. Update queries of the form $(\mathsf{update}, f_1, \ldots, f_n)$ where $|f_i(sk_i, R)| \leq \rho_U(|sk_i| + |R|)$ for any R. If $L_i + |f(sk_i, R)| \leq \rho_U |sk_i|$ holds, the challenger chooses $R_1, \ldots, R_n \xleftarrow{\$} \mathsf{Rnd}[\mathsf{Update}]$ randomly, computes $sk_i \leftarrow \mathsf{Update}_{pk}(sk_i; R_i)$, sends $(f_i(sk_i, R_i))_{i=1}^n$ back to $\mathcal{A}$, and sets $L_i \leftarrow |f_i(sk_i, R_i)|$. Otherwise, the challenger aborts.
- Memory leak queries (leak, f, i), where f is a circuit. If $L_i + |f(sk_i)| \leq \rho_M |sk_i|$ holds, the challenger sends $f(sk_i)$ to adversary and resets $L_i \leftarrow L_i + |f(sk_i)|$. Otherwise, the challenger aborts.
- Signing queries (sig, M, f, i) where f is a circuit with $|f(sk_i, R)| \leq \rho_S(|sk_i| + |R|)$ for any (sk_i, R). The challenger chooses $R \leftarrow \mathsf{Rnd}[\mathsf{Sig}]$ randomly, computes $\sigma \leftarrow \mathsf{Sig}(sk_i, M; R)$ and sends $(\sigma, f(sk_i, R))$ back to $\mathcal{A}$.

Challenge Assuming the challenger did not abort, $\mathcal{A}$ outputs (M_*, σ_*). It succeeds if $\mathsf{Ver}(pk, M_*, \sigma_*) = 1$ holds and $\mathcal{A}$ never made query (sig, M_*, i) for any i.

Definition 1 Let ρ_G, ρ_U, ρ_M, and ρ_S be elements of the real range $[0, 1]$. We say that $\mathcal{SGN} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Update})$ is $(\rho_G, \rho_U, \rho_M, \rho_S)$- *EU-CMA-CML secure* (stand for existentially unforgeable under chosen message attack in the CML model) if no PPT adversary $\mathcal{A}$ succeeds in the experiment of $\mathsf{ExpMLSIG}$ with non-negligible probability. Here $\mathsf{Rnd}[\mathsf{Algo}]$ denote the set of randomnesses for algorithm Algo.

In the full version of this paper [21] we show several negative results regarding generic transformations of single to multi location signature schemes, as well as a simple transformation that does work, under some restrictions on the base signature scheme. We now turn to a direct construction that achieves optimal leakage bounds.

Direct Multi-Location Leakage Resilience The simple generic transformation (described in [21]) may not be satisfactory if the number of locations is very large. For instance, for a key of length 256 bits, even an optimally leakage resilient scheme that is transformed to a multi-location setting with more than 256 locations would be able to withstand less than one bit of leakage per location before the key has to be refreshed. This would require an extremely high refresh rate if even a small (but unknown) number of locations are suspected to leak information.

To address this, we turn to constructing signature and encryption schemes directly, that will withstand large amounts of leakage per location, and will allow the total amount of leakage among different locations to exceed the length of the key between updates. At the core of our constructions is a strengthening of the Leakage Resilient Subspaces Lemma from [5]. On a high level, the BKKV

lemma can be described as follows: let $\mathcal{K}$ be an n-dimensional vector space, and let $Z_1, \ldots, Z_l$, $1 \leq l < k$ be random elements in $\mathcal{K}$. Then, no adversary (even a computationally unbounded one) can distinguish between leakage from random samples from $\mathsf{Span}(Z_1, \ldots, Z_l)$ and random samples from $\mathcal{K}$. The key difference between the lemma in [5] and the one we present here is the ability of the adversary to leak on samples in *parallel* rather than sequentially: we show that even if the adversary breaks his leakage on a given sample into several rounds, where at each round he chooses the leakage function adaptively based on leakage from other samples, he is still unable to distinguish between random samples from $\mathcal{K}$ and from the l dimensional subspace. We next describe the parallel leakage resilient subspace game:

Parallel-leakage resilient subspaces. Let $b \in \{0,1\}$, n, ℓ, m, λ be integers satisfying $n \geq \ell > m \geq 2$, p be a prime, and $\mathcal{K}$ be a n-dimensional vector space over $\mathbb{Z}_p$. The following is the parallel leakage resilient subspace game, played with a computationally unbounded adversary $\mathcal{D}$:

1. Let $Z_1, \ldots, Z_\ell \xleftarrow{\$} \mathcal{K}$. Initially a set $\Gamma = \{\Gamma_1, \ldots, \Gamma_m\}$ of size m is sampled uniformly at random from $\mathcal{K}$ if $b = 1$ and from $\mathsf{Span}(Z_1, \ldots, Z_\ell)$ if $b = 0$.
2. The adversary can make leakage queries: (leak, i, F) where $i \in [m]$ and $F : \mathcal{K} \to \{0,1\}^{\lambda_F}$, $\mathcal{P}_F \subseteq \mathcal{P}$; and refresh queries: $\mathsf{refresh}$. For a leakage query, the adversary is given $F(\Gamma_i)$, as long as $\sum_F \lambda_F \leq \lambda$ where the sum is over all the leakage functions F that are applied to Γ_i between two refresh queries. When the adversary submits a $\mathsf{refresh}$ query, $(\Gamma_1, \ldots, \Gamma_m)$ are assigned a random values from $\mathcal{K}$ if $b = 1$ and random values from $\mathsf{Span}(Z_1, \ldots, Z_\ell)$ if $b = 0$.
3. Finally, $\mathcal{D}$ is given $Z_1, \ldots, Z_\ell$, and it outputs a bit b'.

We denote by $\mathsf{ExpLRS}(b, \mathcal{D})$ the above experiment with an adversary $\mathcal{D}$, and with the bit b specified as a parameter. The output of $\mathsf{ExpLRS}(b, \mathcal{D})$ is defined to be the output of $\mathcal{D}$ at the end of the experiment. We now state the central parallel leakage resilient subspaces lemma:

Lemma 1. *Let $\mathcal{D}$ be an adversary for the above game. Then, for all $\delta \geq 0$, if $2^{\lambda_{\mathsf{total}}} \leq p^{\ell-m-1}\delta^2/q^2$, then*

$$|\Pr[\mathsf{ExpLRS}(0, \mathcal{D}) = 0] - \Pr[\mathsf{ExpLRS}(1, \mathcal{D}) = 0]| \leq \delta.$$

The proof of Lemma 1 appears in the full version of this paper [21].

2.2 Construction

We present a simple adaptation of the signature scheme of [28] to the multi-location setting. The modified construction allows us to achieve optimal leakage resilience, even when multiple versions of the key leak simultaneously. That is, the total amount of leakage across all locations between updates significantly exceeds the length of a single complete key (this is in contrast to the simple

generic transformation, where the amount of leakage per location decreases as the number of locations increases to guarantee that the total amount does not exceed the size of a key). The changes required to the scheme and analysis are quite minimal. Indeed, the only substantial modification to the analysis is the use of the parallel leakage-resilient subspaces lemma (Lemma 1). For completeness, we describe the complete scheme here, and give a high level overview of the necessary modifications to the security analysis.

Our construction relies on the Symmetric External DDH assumption in bilinear groups (details of the assumption are given in [21]). The description of our scheme is as follows. Let $n \geq 3$ and m be integers. Let Setup be a polytime algorithm that generates a group description $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$, as discussed above, where $\mathsf{e} : \mathbb{G} \times \mathbb{H} \to \mathbb{T}$. For $\mathcal{H} = (\boldsymbol{H}_0, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_m) \in (\mathbb{H}^2)^{m+1}$ and $M \in \{0,1\}^m$, we define a Water's hash function [38] h as

$$h_{gk}(\mathcal{H}, M) = \boldsymbol{H}_0 + \sum_{k \in [m]} M_k \boldsymbol{H}_k,$$

where M_k is the k-th bit of M. Let Prf and Vrf be the proof algorithm and the verification algorithm of the Groth-Sahai proof system (reviewed in [21]). Our signature scheme $\mathcal{SGN} = (\mathsf{Kg}, \mathsf{Update}, \mathsf{Sig}, \mathsf{Ver})$ works as follows.

Key Generation $\mathsf{Gen}(1^\kappa, m)$**:** $gk \leftarrow (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e}) \leftarrow \mathsf{Setup}(1^\kappa)$, $\boldsymbol{G} \leftarrow \mathbb{H}^2$, $\mathcal{H} \leftarrow (\boldsymbol{H}_0, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_m) \leftarrow (\mathbb{H}^2)^{m+1}$.
Randomly select $A \xleftarrow{\$} \mathbb{G}$, $Q \xleftarrow{\$} \mathbb{H}$, and $\boldsymbol{a}, \boldsymbol{q} \xleftarrow{\$} \mathbb{Z}_p^n$ satisfying $\langle \boldsymbol{a}, \boldsymbol{q} \rangle = 0$ and compute $\boldsymbol{A} \leftarrow \boldsymbol{a}A$, $\boldsymbol{Q} \leftarrow \boldsymbol{q}Q$. Select $\boldsymbol{W}^{[0]} \xleftarrow{\$} \mathbb{H}^n$ randomly, compute $T \leftarrow \mathsf{e}(\boldsymbol{A}, \boldsymbol{W}^{[0]})$. Then, the location specific keys are generated as: choose $s_i \xleftarrow{\$} \mathbb{Z}_p$, and set $\boldsymbol{W}_i^{[0]} \leftarrow \boldsymbol{W}^{[0]} + s_i \boldsymbol{Q}$. Outputs $pk \leftarrow (gk, \boldsymbol{G}, \mathcal{H}, \boldsymbol{A}, T, \boldsymbol{Q})$ and location specific private keys $(sk_i^{[0]})_{i \in [m]}$.

Key Update $\mathsf{Update}_{pk}(sk^{[i]})$**:** Parse pk and $sk^{[i]}$ as $(gk, \boldsymbol{G}, \mathcal{H}, \boldsymbol{A}, T, \boldsymbol{Q})$ and $\boldsymbol{W}^{[i]}$ respectively, select $s \xleftarrow{\$} \mathbb{Z}_p$ randomly, and output $sk^{[i+1]} \leftarrow \boldsymbol{W}^{[i+1]} \leftarrow \boldsymbol{W}^{[i]} + s\boldsymbol{Q}$.

Signing $\mathsf{Sig}(sk^{[i]}, M)$ **for** $M \in \{0,1\}^m$**:** Parse pk and $sk^{[i]}$ as $(gk, \boldsymbol{G}, \mathcal{H}, \boldsymbol{A}, T, \boldsymbol{Q})$ and $\boldsymbol{W}^{[i]}$. Compute $\boldsymbol{H}_M \leftarrow h_{gk}(\mathcal{H}, M)$, set $crs_M \leftarrow (\boldsymbol{G}, \boldsymbol{H}_M)$, and $\sigma \leftarrow \mathsf{Prf}(gk, crs_M, (\boldsymbol{A}, T), \boldsymbol{W}^{[i]})$ and output σ.

Verification $\mathsf{Ver}(pk, M, \sigma)$**:** Parse pk as $(gk, \boldsymbol{G}, \mathcal{H}, \boldsymbol{A}, T, \boldsymbol{Q})$, compute $\boldsymbol{H}_M \leftarrow h_{gk}(\mathcal{H}, M)$, and set $crs_M \leftarrow (\boldsymbol{G}, \boldsymbol{H}_M)$. If $\mathsf{Ver}(gk, crs_M, (\boldsymbol{A}, T), \sigma) = 1$, output 1. Otherwise, output 0.

Theorem 2. *For any constants $c > 0$ and any $\gamma = \Theta(1/\sqrt{\kappa})$, the proposed scheme $\mathcal{SIG}$ is $(\rho_G, \rho_U, \rho_M, \rho_S)$-EU-CMA-CML secure under the SXDH assumption. Here*

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

We can achieve the fraction $1 - o(1)$ of leakage in signing and in memory by setting $n = \kappa$.

Overview of the modifications to the analysis of the MTVY scheme. Essentially the analysis of the above scheme proceeds similarly to the analysis of the original (single-location) variant of [28]. The main modification to the argument is to replace the use of the original leakage resilient subspace lemma from [5] with our parallel version, given by lemma 1. Specifically, in the proof of [28, Lemma 9] we replace the use of [28, Proposition 10] with our Lemma 1. Once we use Lemma 1, the subspace $\mathcal{W}$ remains information theoretically hidden from the adversary throughout the security, and therefore any successful forgery would yield a successful attack on the Independent Pre-Image Resistant Hash Function described in [28, Section 3]. We leave the full details of the analysis to the full version of this paper.

3 Authenticated session protocols

Next, we describe our definition and construction of a leakage resilient authenticated session protocol in the private key setting.

3.1 Security definition

The intuitive goal of an authenticated session protocol involving two parties, the sender A and the receiver B, where A is sending message pieces $m_1, m_2, \ldots,$ to B, is that B can verify that the message pieces he receives are indeed those sent by A, in the same order. This should hold even when all message pieces m_i sent by A are adversarially chosen. Of course, the adversary has complete control of the public channel over which A and B are communicating. This means that he controls the timing and contents of all communication. In the leakage-resilient case, we strengthen the adversary by allowing him to obtain leakage on *both* parties. We are interested in the *continual* leakage setting, where the adversary obtains some bounded amount of leakage on each computation by each party but the total amount of leakage obtained by the adversary over the course of the execution of the protocol is unbounded. The leakage on each computation is computed by an adversarially-chosen function is applied to the inputs and randomness involved in the computation along with the *entire* state of the party performing the computation. This means that we do *not* rely on the only-computation-leaks assumption. In our case, we further strengthen the adversary by giving him all the entropy used by each party after the initial state. Equivalently, we require that A and B are deterministic but each have access to a (separate) source of *public min-entropy*; whenever a party obtains a string its source of high min-entropy strings, this string is also given to the adversary.

We begin by formally defining session protocols (we restrict our definition to protocols as ours with two flows per message, but the idea can be extended).

Definition 3 (Shared-private-key session protocol with public min-entropy) *A* shared-private-key session protocol with public min-entropy *(which we will henceforth simply refer to as a* session protocol*) consists of deterministic poly-time algorithms* EvalB_1 *(producing message from B),* EvalA *(receiving the emssage*

from B), and EvalB_2 *(producing the message received from after evaluation), polynomials $s_B(n)$, $\ell_B(n)$, $s_A(n)$, and $\ell_A(n)$, and distribution ensembles $\{Z_n^A\}$ and $\{Z_n^B\}$ that satisfy the following properties for all $n \in \mathbb{N}$:*

1. *Z_n^A is a distribution over strings of length $s_A(n)$ such that $\mathbf{H}_\infty(Z_n^A) \geq \log^2(n)$. Similarly, Z_n^B is a distribution over strings of length $s_B(n)$ such that $\mathbf{H}_\infty(Z_n^B) \geq \log^2(n)$.*

2. EvalB_1 *takes as input $K_B \in \{0,1\}^n$ and $r_B \in \{0,1\}^{s_B(n)}$, and outputs $\beta \in \{0,1\}^{\ell_B(n)}$ and $K'_B \in \{0,1\}^n$ such that β has prefix r_B.*

   Informally, the strings K_B and K'_B are the state of party B before and after it executes EvalB_1*, r_B is the public min-entropy used by* EvalB_1*, and β is a flow from party B to party A.*

3. EvalA *takes as input $K_A \in \{0,1\}^n$, $m \in \{0,1\}^n$, $\beta \in \{0,1\}^{\ell_B(n)}$, and $r_A \in \{0,1\}^{s_A(n)}$, and outputs $e \in \{0,1\}^{\ell_A(n)}$ and $K'_A \in \{0,1\}^n$ such that e has prefix r_A.*

   Informally, the strings K_A and K'_A are the state of party A before and after it executes EvalA*, m is a message piece that party A would like to send to party B, β is a flow from party B to party A, r_A is the public min-entropy used by* EvalA*, and e is a flow from party A to party B.*

4. EvalB_2 *takes as input $K_B \in \{0,1\}^n$, $r_B \in \{0,1\}^{s_B(n)}$, and $e \in \{0,1\}^{\ell_A(n)}$, and outputs either $m \in \{0,1\}^n$ and $K'_B \in \{0,1\}^n$ or a special message* `Fail`*.*

   Informally, the strings K_B and K'_B are the state of party B before and after it executes EvalB_2*, r_B is the public min-entropy used by the immediately preceding run of* EvalB_1*, e is a flow from party A to party B, and m is a message piece received by party B.*

5. *For all $K \in \{0,1\}^n$, every polynomial $p(n)$, all $r_{A,1}, r_{A,2}, \ldots, r_{A,p(n)} \in \{0,1\}^{s_A(n)}$, all $r_{B,1}, r_{B,2}, \ldots, r_{B,p(n)} \in \{0,1\}^{s_B(n)}$, and all sequences of message pieces $m_1, m_2, \ldots, m_{p(n)} \in \{0,1\}^n$, if we define $K_{A,0} = K_{B,0} = K$ and, for $1 \leq i \leq p(n)$, we iteratively define $K_{A,i}, K'_{B,i}, K_{B,i}, e_i, \beta_i, m'_i$ in the following manner:*

$$\begin{aligned}(\beta_i, K'_{B,i}) &\leftarrow \mathsf{EvalB}_1(K_{B,i-1}, r_{B,i}) \\ (e_i, K_{A,i}) &\leftarrow \mathsf{EvalA}(K_{A,i-1}, m_i, \beta_i, r_{A,i}) \\ (m'_i, K_{B,i}) &\leftarrow \mathsf{EvalB}_2(K'_{B,i}, r_{B,i}, e_i)\end{aligned}$$

   then $m'_i = m_i$ for all $1 \leq i \leq p(n)$.

   Informally, this means that in the absence of an adversary, the message pieces output by party B are exactly those sent by party A, in the same order.

We now define the security experiment for leakage-resilient authenticated session protocols. The adversary will be a family of polynomial-size circuits $C = \{C_n\}$. Letting $\lambda : \mathbb{N} \to \mathbb{N}$ be a function, we will say that an adversary C is

$\lambda(n)$-bounded if the leakage functions produced by C_n over the course of the security experiment each have output length $\lambda(n)$. Fixing a session protocol

$$(\mathsf{EvalB}_1, \mathsf{EvalA}, \mathsf{EvalB}_2, s_B(n), \ell_B(n), s_A(n), \ell_A(n), \{Z_n^A\}, \{Z_n^B\})$$

a function $\lambda : \mathbb{N} \to \mathbb{N}$, a $\lambda(n)$-bounded adversary $C = \{C_n\}$, and $n \in \mathbb{N}$, the security experiment proceeds as follows.

A string $K \in \{0,1\}^n$ is randomly chosen. We define $K_{A,0} = K_{B,0} = K$. Then, C_n is allowed to run EvalA, EvalB_1, and EvalB_2 in the following manner. C_n may run these algorithms as many times as he wishes and in any order of his choice as long as for every $i > 0$, the $(i+1)$-st invocation of EvalB_1 does not occur before the i-th invocation of EvalB_2, and the i-th invocation of EvalB_2 does not occur before the i-th invocation of EvalB_1. (This restriction captures the fact that even though the adversary controls the public channel, party B will still alternate between executing EvalB_1 and executing EvalB_2.) We now describe what happens when the adversary C_n runs each algorithm.

- For $i > 0$, the i-th invocation of EvalB_1 proceeds as follows. C_n produces the description of a circuit $f_{B1,i} : \{0,1\}^n \times \{0,1\}^{s_B(n)} \to \{0,1\}^{\lambda(n)}$. Then, $r_{B,i} \leftarrow Z_n^B$ is chosen. Next, $(\beta_i, K'_{B,i}) \leftarrow \mathsf{EvalB}_1(K_{B,i-1}, r_{B,i})$ and $leak_{B1,i} \leftarrow f_{B1,i}(K_{B,i-1}, r_{B,i})$ are computed. Finally, C_n is given β_i and $leak_{B1,i}$.
- For $i > 0$, the i-th invocation of EvalA proceeds as follows. C_n produces $m_i \in \{0,1\}^n$ and $\beta'_i \in \{0,1\}^{\ell_B(n)}$, and the description of a circuit $f_{A,i} : \{0,1\}^n \times \{0,1\}^{s_A(n)} \to \{0,1\}^{\lambda(n)}$. Then, $r_{A,i} \leftarrow Z_n^A$ is randomly chosen. Next, $(e_i, K_{A,i}) \leftarrow \mathsf{EvalA}(K_{A,i-1}, m_i, \beta'_i, r_{A,i})$ and $leak_{A,i} \leftarrow f_{A,i}(K_{A,i-1}, r_{A,i})$ are computed[4]. Finally, C_n is given e_i and $leak_{A,i}$.
- For $i > 0$, the i-th invocation of EvalB_2 proceeds as follows. C_n produces a string $e'_i \in \{0,1\}^{\ell_A(n)}$ and the description of a circuit $f_{B2,i} : \{0,1\}^n \to \{0,1\}^{\lambda(n)}$. Then, $(m'_i, K_{B,i}) \leftarrow \mathsf{EvalB}_2(K'_{B,i}, r_{B,i}, e'_i)$ and $leak_{B2,i} \leftarrow f_{B2,i}(K'_{B,i})$ are computed[5]; if EvalB_2 outputs `Fail`, the experiment ends immediately. If the i-th invocation of EvalA has previously occurred and $m'_i = m_i$, C_n is given $leak_{B2,i}$; otherwise, the experiment ends immediately.

Say that the final invocation of EvalB_2 is the j-th invocation. Define $q_C(n)$ to be the probability that the j-th invocation of EvalB_2 does not output `Fail` and either EvalA has been invoked fewer than j times or $m'_j \neq m_j$.

Definition 4 (Leakage-resilient authenticated session protocol) *Let $\lambda : \mathbb{N} \to \mathbb{N}$ be a function. A session protocol is a* $\lambda(n)$-leakage-resilient authenticated session protocol *if for every $\lambda(n)$-bounded adversary C as above, we have $q_C(n) \leq 1/n^d$ for all d and sufficiently large n.*

[4] It is not necessary to provide m_i or β'_i as inputs to $f_{A,i}$ since C_n chose these values himself and hence he can simply hardcode them into $f_{A,i}$ if he wishes.

[5] It is not necessary to provide $r_{B,i}$ to $f_{B2,i}$ since this was previously provided to C_n as the prefix of β_i, and it is not necessary to provide e'_i to $f_{B2,i}$ since C_n chose this value himself.

3.2 Our construction

In our construction, only party B requires a source of public min-entropy. Accordingly, to simplify notation, we use Z_n rather that Z_n^B to denote the high min-entropy distribution used by B.

Given pseudo-random function generators $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ and $F' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, and given a distribution ensemble $\{Z_n\}$ such that for all n, Z_n is a distribution over $\{0,1\}^n$ and $\mathbf{H}_\infty(Z_n) \geq \log^2(n)$, we construct a leakage-resilient authenticated session protocol SP as follows.

EvalB_1**:** On input (K_B, r_B), where $K_B \in \{0,1\}^n$ and $r_B \in \{0,1\}^n$, EvalB_1 lets $K'_B = K_B$ and $\beta = r_B$, and outputs (β, K'_B).

EvalA**:** On input (K_A, m, β), where $K_A, m \in \{0,1\}^n$ and $\beta \in \{0,1\}^n$, EvalA computes $K'_A||X_A \leftarrow F_{K_A}(\beta)$ (where $|K'_A| = |X_A| = n$) and $\alpha = F'_{X_A}(m)$, lets $e = \langle m, \alpha \rangle$, and outputs (e, K'_A).

EvalB_2**:** On input (K_B, r_B, e'), where $K_B \in \{0,1\}^n$, $r_B \in \{0,1\}^n$, and $e' \in \{0,1\}^{2n}$, EvalB_2 parses $\langle m', \alpha' \rangle \leftarrow e'$, computes $K'_B||X_B \leftarrow F_{K_B}(r_B)$ (where $|K'_B| = |X_B| = n$), and $\alpha = F'_{X_B}(m')$. If $\alpha' = \alpha$, EvalB_2 outputs (m', K'_B); otherwise, EvalB_2 outputs `Fail`.

It is not hard to see that SP satisfies the definition of a session protocol. The idea is that parties A and B both run a stream cipher (see Section 4) starting from the same key and using the same inputs, and use the i-th output X_i to compute a signature $F'_{X_i}(m_i)$ of the i-th message piece m_i.

Theorem 5. *For all $c > 0$, SP is a $c \log n$-leakage-resilient authenticated session protcol.*

For the details of the proof of Theorem 5 we direct the reader to [21].

4 Stream cipher construction

In this section, we present our modified version of Pietrzak's stream cipher. The main purpose of our construction is to prove Theorem 6, which in turn is used in the proof of Theorem 5. Our construction uses only a single piece of memory but requires a public source of min-entropy. We believe that the construction below and its analysis are of independent interest due to the involved analysis of the leakage resilient stream cipher with public randomness.

The construction. Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudo-random function. Let $\{Z_n\}$ be such that for all n, Z_n is a distribution over strings of length n and $\mathbf{H}_\infty(Z_n) \geq \log^2(n)$. The initial state is K_0, where $K_0 \in \{0,1\}^n$ is randomly chosen. For each $i > 0$, the i-th round consists of:

1. $R_i \leftarrow Z_n$ is chosen.
2. $K_i||X_i \leftarrow F_{K_{i-1}}(R_i)$.
3. The new state is K_i.

The adversary's interaction. Fix $c > 0$. A $(c \log n)$-bounded adversary interacts as follows. For each $i > 0$:

1. Before round i, the adversary outputs the description of a function $f_i : \{0,1\}^{2n} \to \{0,1\}^{c \log n}$.
2. After round i, the adversary sees $R_i, X_i, f_i(K_{i-1}, R_i)$.

4.1 Security analysis

We begin by defining some notation.

For an adversary A, we will use $\texttt{real}_i$ to denote the adversary's view after the first i rounds along with the corresponding X_j. That is,

$$\texttt{real}_i = \langle R_1, f_1(K_0, R_1), X_1, R_2, f_2(K_1, R_2), X_2, \ldots, R_i, f_i(K_{i-1}, R_i), X_i \rangle$$

Note that the f_j are not fixed functions, but rather are chosen adaptively by the adversary A as described in Section 4.

We will also define a version of $\texttt{real}_i$ that includes an additional round where there is no leakage. Specifically, we define

$$\texttt{real}_i^+ = \langle \texttt{real}_i, R_{i+1}, K_{i+1}, X_{i+1} \rangle$$

That is, $\texttt{real}_i^+$ includes the inputs and outputs of an additional leak-free round along with the entire state at the end of that round.

Theorem 6. *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudo-random function. Let K' and X' be independent random variables that are each uniformly distributed over $\{0,1\}^n$. Let $\{Z_n\}$ be such that for all n, Z_n is a distribution over strings of length n and $\mathbf{H}_\infty(Z_n) \geq \log^2(n)$. For all $c > 0, d > 0, e > 0$, every function $p : \mathbb{N} \to \mathbb{N}$, sufficiently large n, all $(c \log n)$-bounded adversaries A interacting as described in section 4 and obtaining leakage for $p(n)$ rounds, and all adversaries D such that $2 \cdot \texttt{size}(A) + \texttt{size}(D) + p(n)\texttt{size}(F) \leq n^e$,*

$$\left| \Pr\left[D(\texttt{real}^+_{p(n)}) = 1 \right] - \Pr\left[D(\texttt{real}_{p(n)}, R_{p(n)+1}, K', X') = 1 \right] \right| \leq \frac{6p(n) + 6}{n^d}.$$

Specifically, for sufficiently large n (depending only on c, d, and e), if there exists an adversary D breaking the above, then there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$.

The details of the proof of Theorem 6 are given in [21]. The high-level approach is similar to that of Pietrzak [31], but there are differences in the details, due to the differences in our security models.

References

1. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, March 2009.

2. Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 113–134. Springer, May 2010.
3. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, August 2009.
4. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 89–108, Berlin, Heidelberg, 2011. Springer-Verlag.
5. 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 Annual Symposium on Foundations of Computer Science*, pages 501–510. IEEE Computer Society Press, 2010.
6. Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: Built-in tamper resilience. Cryptology ePrint Archive, Report 2010/503, 2010. `http://eprint.iacr.org/`. To appear in ASIACRYPT 2011.
7. Sherman S. M. Chow, Yevgeniy Dodis, Yannis Rouselakis, and Brent Waters. Practical leakage-resilient identity-based encryption from simple assumptions. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 152–161. ACM Press, October 2010.
8. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st Annual Symposium on Foundations of Computer Science*, pages 511–520. IEEE Computer Society Press, 2010.
9. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, December 2010.
10. Yevgeniy Dodis, Allison Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. Cryptology ePrint Archive, Report 2011/369, 2011. `http://eprint.iacr.org/`. To appear in FOCS 2011.
11. Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, August 2010.
12. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302. IEEE Computer Society Press, October 2008.
13. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, February 2010.
14. Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, May 2010.

15. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, February 2004.
16. Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 59–79. Springer, August 2010.
17. J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.
18. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, May / June 2006.
19. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, August 2003.
20. Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 41–58. Springer, August 2010.
21. Ali Juma, Yevgeniy Vahlis, and Moti Yung. Multi-location leakage resilient cryptography. Cryptology ePrint Archive, March 2012. `http://eprint.iacr.org/`.
22. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, December 2009.
23. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, August 1996.
24. Yuichi Komano, Kazuo Ohta, Hideyuki Miyake, and Atsushi Shimbo. Algorithmic tamper proof (ATP) counter units for authentication devices using PIN. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09: 7th International Conference on Applied Cryptography and Network Security*, volume 5536 of *Lecture Notes in Computer Science*, pages 306–323. Springer, June 2009.
25. Allison Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. Cryptology ePrint Archive, Report 2010/562, 2010. `http://eprint.iacr.org/`.
26. Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.
27. Feng-Hao Liu and Anna Lysyanskaya. Algorithmic tamper-proof security under probing attacks. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 106–120. Springer, September 2010.
28. Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC '11: Proceedings of the 8th Theory of Cryptography Conference*, 2011.

29. Paulo Mateus and Serge Vaudenay. On tamper-resistance from a theoretical viewpoint. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 411–428. Springer, September 2009.
30. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.
31. Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, April 2009.
32. Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09: 16th Conference on Computer and Communications Security*, pages 199–212. ACM Press, November 2009.
33. Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 210–218. Springer, January 1997.
34. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.
35. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, April 2009.
36. Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient cache attacks on AES, and countermeasures. *Journal of Cryptology*, 23(1):62–74, January 2010.
37. Nicolas Veyrat-Charvillon and François-Xavier Standaert. Generic side-channel distinguishers: Improvements and limitations. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 354–372. Springer, 2011.
38. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.
39. Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 141–151. ACM Press, October 2010.