Cryptography with Tamperable and Leaky Memory

Yael Tauman Kalai[1], Bhavana Kanukurthi[2,⋆], and Amit Sahai[3,⋆⋆]

[1] Microsoft Research
[2] Boston University
[3] University of California (UCLA)

Abstract. A large and growing body of research has sought to secure cryptographic systems against physical attacks. Motivated by a large variety of real-world physical attacks on memory, an important line of work was initiated by Akavia, Goldwasser, and Vaikuntanathan [1] where security is sought under the assumptions that: (1) *all memory is leaky*, and (2) leakage can be *an arbitrarily chosen (efficient) function* of the memory.

However, physical attacks on memory are not limited to leakage through side-channels, but can also include active *tampering* attacks through a variety of physical attacks, including heat and EM radiation. Nevertheless, protection against the analogous model for tampering – where (1) *all memory is tamperable*, and (2) where the tampering can be *an arbitrarily chosen (efficient) function* applied to the memory – has remained an elusive target, despite significant effort on tampering-related questions.

In this work, we tackle this question by considering a model where we assume that *both* of these pairs of statements are true – that all memory is both leaky and (arbitrarily) tamperable. Furthermore, we assume that this leakage and tampering can happen repeatedly and continually (extending the model of [10, 7] in the context of leakage). We construct a signature scheme and an encryption scheme that are provably secure against such attacks, assuming that memory can be updated in a randomized fashion between episodes of tampering and leakage. In both schemes we rely on the linear assumption over bilinear groups.

We also separately consider a model where only continual and repeated tampering (but only bounded leakage) is allowed, and we are able to obtain positive results assuming only that "self-destruct" is possible, without the need for memory updates.

Our results also improve previous results in the continual leakage regime without tampering [10, 7]. Whereas previous schemes secure against continual leakage (of arbitrary bounded functions of the secret key), could tolerate only $1/2 - \epsilon$ leakage-rate between key updates under the linear assumption over bilinear groups, our schemes can tolerate $1 - \epsilon$ leakage-rate between key updates, under the same assumption.

⋆ Research supported in part by CNS-0546614, CNS-0831281, CNS-1012910. Some of this work was done while visiting Microsoft Research, New England.

⋆⋆ Research supported in part from a DARPA/ONR PROCEED award, NSF grants 0916574 and 0830803, a Xerox Foundation Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant.

1 Introduction

A large and growing body of research has sought to secure cryptographic systems against physical attacks (e.g. [8, 23, 28, 18, 22, 13, 31, 1, 29, 25, 11, 9, 15, 24, 19]). Motivated by a large variety of real-world physical attacks on memory [26, 27, 30, 21, 20], an important line of work was initiated by Akavia, Goldwasser, and Vaikuntanathan [1] where security is sought under the assumptions that: (1) *all memory is leaky*, and (2) leakage can be *an arbitrarily chosen (efficient) function* of the memory[4].

However, physical attacks on memory are not limited to leakage through side-channels, but can also include active *tampering* attacks (see [20] and the references therein) through a variety of physical attacks, such as exposure to heat and EM radiation [17, 33, 32]. Nevertheless, protection against the analogous model for tampering – where (1) *all memory is tamperable*, and (2) where the tampering can be *an arbitrarily chosen (efficient) function* applied to the memory – has remained an elusive target, despite significant effort on tampering-related questions (e.g. [18, 22, 3, 14, 2])[5]. In this work, we tackle this question by considering a model where we assume that *both* of these pairs of statements are true – that all memory is both leaky and (arbitrarily) tamperable. Furthermore, we assume that this leakage and tampering can happen repeatedly and continually (extending the model of [10, 7] in the context of leakage). We show strong positive results for cryptographic tasks including signatures and decryption, assuming that memory can be updated in a randomized fashion between episodes of tampering and leakage. We note that we only consider tampering with the memory and do not consider the question of tampering during the computation (which might occur, for instance, due to fault attacks [5, 4, 27]).

We also separately consider a model where only continual and repeated tampering (but bounded leakage) is allowed, and we are able to obtain positive results assuming only that "self-destruct" is possible, without the need for memory updates.

Background. Before explaining our results in greater detail, let us first recall the results of [10, 7], which we strengthen. These results construct various cryptographic schemes (such as encryption and signature schemes), that remain secure even if the secret key is being continually leaked. In this continual leakage model, at each time period the adversary can make a *leakage query* L, where L is a poly-size circuit, and get back $L(sk)$. Clearly, in order to get security under continual leakage, one must bound the leakage function L in some way, since if, for example, L is the identity function, security is clearly breached. Both works [10, 7] allow L to be any *shrinking* leakage function; *e.g.*, any L such that $|L(sk)| \leq 0.99|sk|$. More generally, they allow any leakage function L such that sk has "enough" min-entropy left conditioned on $L(sk)$. We note that several

[4] To prevent trivial attacks, the leakage function should either be significantly shrinking or leave the memory with enough entropy.

[5] We elaborate on these related works in Section 1.3.

earlier continual leakage models were considered in the literature, such as the one due to Micali and Reyzin [28] who consider the "only computation leaks information" assumption, and the one due to Ishai, Sahai and Wagner [23], who consider only leakage of individual bits produced during honest computation. We elaborate on these related works, as well as others, in Section 1.3.

As was argued in [7, 10], since the leakage may be continual, and at any time period the adversary can learn *any* bounded poly-size function of the secret key, the secret-key must be periodically *updated*, since otherwise it will eventually be completely leaked. We emphasize that this should be done without modifying the public key, and these updates should be oblivious to all other users. In addition, *deletions* must be allowed, since otherwise the adversary can choose to (gradually) leak the initial secret-state from which everything can be derived. In such case, updating the state is useless because the adversary will leak from the original state and not the updated one.

We note that both these results rely on the linear assumption over bilinear groups. The work of [7] allow $1/2 - \epsilon$ of the secret key to leak during each time period, whereas [10] allow only $1/3 - \epsilon$ of the secret key to leak during each time period.[6] We note that in addition to our main results concerning tampering, we improve the leakage bounds of [10, 7], even if we restrict our attention only to the regime of continual leakage (as opposed to continual leakage and tampering).

1.1 Our CTL Model

In this work, we extend the continual leakage model of [7, 10], and consider not only continual leakage attacks, but also continual *tampering* attacks. Namely, we consider an adversary, that uses side channel attacks not only to continually leak information about the secret key, but also to continually tamper with the secret key. For example, consider an adversary that is given a signature card. The adversary may use various side channel attacks, such as timing attacks and power attacks, to continually *leak* information about the secret key stored in the card, but may also use various other physical attacks [20] to *tamper* with this secret key. For example, the adversary can extract information by causing mutations in the secret key, and then observing the input/output behavior of the tampered system (i.e., observing signatures of messages with respect to tampered secret keys).

More formally, our model generalizes the continual leakage model of [10, 7], in that we allow the adversary at each time period to make any bounded leakage query L, *as well as any tampering query* T. Both types of queries are modeled as a poly-size circuit. After the adversary makes a tampering query T, the secret key sk is replaced with $T(sk)$. We call this model the *continual tampering and leakage* model, or the CTL model, for short. We note that we only allow tampering with the secret key and not the actual computation. We say that a scheme is secure in the CTL model if after continually leaking information about the secret keys,

[6] We note that under the less standard SXDH assumption their leakage rate can increase to $1 - \epsilon$ in [7] and $1/2 - \epsilon$ in [10].

and continually tampering with them, the adversary cannot break security with respect to the *original* secret key; i.e., in the case of signature schemes, the adversary cannot forge signatures with respect to the original verification key; in the case of encryption schemes, the adversary cannot break semantic security with respect to the original public key.

At first glance, the reader may wonder whether tampering attacks can be viewed as leakage attacks, and thus whether the CTL model is at all more general than the continual leakage model. The answer is that it is indeed more general, for the following two reasons. First, in leakage attacks there is always an assumed bound on the leakage size (within each time period, in the case of continual leakage). On the other hand, in a tampering attack, the adversary can mutate the key in an arbitrary manner, and receive signatures with this mutated key. Note that if such signatures reveal the entire mutated secret key (which is possible since the key is malformed), then this tampering attack is an illegal leakage attack (since this leakage is not bounded). The other (and perhaps more profound) reason why the CTL model seems much harder to deal with than the continual leakage model, is that in the CTL model the update procedure updates a *mutated* key, as opposed to an honestly generated key. Thus, we need to argue that the update procedure is still "effective" even if it is applied to mutated keys. We note that the schemes of [10, 7] are not secure in the CTL model.

An additional assumption that we make is that we assume that a CRS is honestly chosen and hard-coded into the cryptographic system, not in memory but into the logic itself, so that the CRS is not tamperable. We stress that the CRS is fixed once and for all and does not depend on any secret information chosen later by the user, which is stored in memory. We note that to some extent, the hardware manufacturer must always be trusted to implement the correct algorithms in hardware. (For instance, we must trust that the manufacturer did not implement an algorithm that leaks secrets through specific subliminal channels.) Thus we are not asking a significantly higher level of trust from the user to assume that the manufacturer (or some outside entity with higher trust) honestly chose a CRS and that the manufacturer correctly hard-coded this CRS into the logic of the device. We leave the problem of eliminating the CRS, or exploring ways to reduce the level of trust needed in the manufacturer, as an important open problem.

1.2 Our Results

In what follows we present informal statements of our results.

Theorem 1. *Under the linear assumption in a prime order group G, there exists a encryption scheme that is semantically secure in the* CTL *model, tolerating a leakage of $1 - \epsilon$ in each time period.*

Theorem 2. *Under the linear assumption in a prime order group G, there exists a signature scheme that is existentially unforgeable under adaptive chosen message attacks in the* CTL *model, tolerating a leakage of $1 - \epsilon$ in each time period.*

Theorem 3. *Given a signature scheme that is existentially unforgeable in the bounded leakage model, there exists an efficient transformation to covert it into a signature scheme that is existentially unforgeable in the continual tampering and bounded leakage model.*[7]

The proofs of these theorems are deferred to the full version.

1.3 Previous Work

Work on tolerating leakage was initiated by Rivest and Boyko [34, 6] in the context of increasing the cost of brute-force attacks on block ciphers and efficiency issues. Ideas there were applied to the context of leakage by numerous works on exposure-resilient cryptography [8, 12, 23]. These works consider simple leakage functions that reveal a subset of the bits of the secret key or the internal memory of the cryptographic device. This line of research culminated in the work of Ishai, Sahai and Wagner [23] who show how to "compile" any cryptographic algorithm into one that tolerates such types of leakage.

In contrast to these works, that consider leakage functions that act locally, the focus of later works has been on more powerful leakage functions that can perform some *global computation* on the secret key. Micali and Reyzin [28] proposed to construct and study *formal models* that capture *general* types of leakage. This study has led to two distinct strands of work, described below.

Bounded Leakage Models. This line of work considers the leakage model that allows the adversary to obtain the output of applying any efficiently computable function f, of his choice, to the secret key sk. This is allowed as long as the output $f(sk)$ "does not reveal the entire secret key". This latter condition has been formalized in many different ways, starting with the work of Akavia, Goldwasser and Vaikuntanathan [1] who restrict the leakage function f to have output length $\ell(|sk|) \ll |sk|$ and subsequently in the works of Naor and Segev [29] and Dodis, Kalai and Lovett [11]. Constructions of cryptosystems satisfying one or more of these definitions can be found in various works [1, 29, 11, 25, 9].

Continual Leakage Models. This line of work considers the case where the leakage is continual, i.e., a bounded amount of information about the secret key is leaked in each time period, but the overall leakage in the entire lifetime of the secret key is unbounded. It is easy to see that to guarantee any security in this model the secret key must necessarily be updated between time-periods. Micali and Reyzin [28] proposed to study security against continual leakage under the assumption that "only computation leaks information". In other words, information leakage may occur any time a computation takes place; however, the assumption is that the parts of memory that are not involved in the computation during a certain time-period, are not subject to leakage during that time-period. A number of works (for example [13, 31, 16, 24, 19]) design cryptographic schemes that are resilient to leakage under this assumption.

[7] For this result we need to assume that the signature card can self-destruct.

Very recently, Dodis, Haralambiev, Lopez-Alt and Wichs [10] and Brakerski, Kalai, Katz, and Vaikuntanathan [7] constructed cryptographic schemes (such as signature schemes and encryption schemes) in the continual leakage model without this assumption that only computation leaks. Our main result builds upon the result of Brakerski *et al.*

Tampering models. Several models of security against tampering have been considered in the past, however all of them differ substantially from ours. The notion of algorithmic tamper-proof security was proposed by Gennaro, Lysyanskaya, Malkin, Micali, and Rabin [18] in which devices have *both* tamper-proof but fully leakable memory (which can be programmed with key information uniquely customized to each user), along with tamperable memory – which can be continually tampered with using arbitrary polynomial-time tampering functions. (In contrast, our model allows all memory to be arbitrarily tampered.) Security against related-key attacks was formalized by Bellare and Kohno [3] in the context of block-cipher security, but can be seen as relating to a model where the key can be tampered with. However, this line of work (see [2] and references therein) deals with limited kinds of tampering functions, such as functions that simply XOR the key with a fixed value or affine linear transformations. The recent work on non-malleable codes [14] also looks as such limited tampering functions. (In contrast, our model allows arbitrary polynomial-time tampering functions.) Finally, the work of Ishai, Prabhakaran, Sahai, and Wagner [22] considers still weaker tampering functions, which only allow individual bits to be toggled or set to a specific value. However, in contrast to our work, Ishai *et al.* also allow tampering to affect the computation, and not just the memory, which is beyond the scope of our work.

2 Overview

Notation. We use the following notational conventions. Bold uppercase denotes matrices ($\mathbf{X} \in \mathbb{Z}_p^{n \times k}$) and bold lowercase denotes vectors ($\mathbf{x} \in \mathbb{Z}_p^n$). All vectors are column vectors, row vectors are denoted by $\mathbf{x}^T$. In what follows, we let $\mathbb{G}$ be a multiplicative group of prime order p, and g be a generator of $\mathbb{G}$. We let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. All our results assume that the linear assumption holds in $\mathbb{G}$.[8] For a matrix $\mathbf{X} \in \mathbb{Z}^{k \times n}$ (or a vector, as a special case), we let $g^{\mathbf{X}}$ denote a $k \times n$ matrix such that $(g^{\mathbf{X}})_{i,j} = g^{(\mathbf{X})_{i,j}}$.

Let X be a probability distribution over a domain S, we write $x \leftarrow X$ to indicate that x is sampled from the distribution X. For a set S, we write $x \leftarrow S$ to indicate that x is chosen uniformly from S. Two ensembles $X = \{X_k\}_k$, $Y = \{Y_k\}_k$ are said to be $\epsilon = \epsilon(k)$-*close* if the statistical distance between them is at most $\epsilon(k)$, this is also denoted by $X \stackrel{\epsilon}{\equiv} Y$.

[8] Formally, $\mathbb{G} = \{\mathbb{G}_k\}_{k \in \mathbb{N}}$ where each group $\mathbb{G}_k$ is a group of prime order p, where p is a k bit prime, and where k is the security parameter.

As was mentioned in the introduction, our work builds on the work of [7], which (among other things) constructs encryption and signature schemes that were proven secure against continual leakage. Let us start by recalling some of the ideas in [7], which are relevant to this work. In both their encryption scheme and their signature scheme, the public key is of the form $g^{\mathbf{a}}$ and the secret key is of the form $g^{\mathbf{b}}$, where g is a generator of $\mathbb{G}$, and $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_p^\ell$ such that $\mathbf{a} \cdot \mathbf{b} = 0$.[9] The secret key is updated by simply multiplying it by a random scalar in the exponent, i.e., updating it to be $g^{\alpha \cdot \mathbf{b}}$ where $\alpha \leftarrow \mathbb{Z}_p$. Thus, the updated secret keys are simply random elements in span($\mathbf{b}$) (in the exponent). To prove security against continual leakage, they rely on the following lemma, which states that random subspaces are resilient to continual leakage.

Lemma 1. *[7] Let $\mathbf{B} \leftarrow \mathbb{Z}_p^{\ell \times d}$ be a random subspace of dimension d where $\ell \geq d \geq 2$. Let $\mathbf{r} \leftarrow \mathbb{Z}_p^d$ and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$. Then for any leakage function $L : \mathbb{Z}_p^\ell \rightarrow W$, chosen independent of $\mathbf{B}$,*

$$(\mathbf{B}, L(\mathbf{B} \cdot \mathbf{r})) \stackrel{\epsilon}{\equiv} (\mathbf{B}, L(\mathbf{u}))$$

as long as $|W| \leq p^{d-1} \cdot \epsilon^2$.

Intuitively, the above lemma says that leakage on random points taken from a random subspace is indistinguishable from leakage on random points taken from the whole space. In other words, leakage on random points of a subspace doesn't reveal *any* information about the subspace.

This lemma is used as follows: First, use the DDH assumption to claim that it is computationally hard to distinguish between the case that all the secret keys are indeed random in span($\mathbf{b}$) or are random in a random dimension-2 subspace $\mathbf{B} \subseteq \ker(\mathbf{a})$.[10] Therefore, if there exists an adversary that breaks security, then this adversary will also succeed in breaking the security if all the secret keys were random elements in $\mathbf{B}$ (in the exponent), as opposed to random elements in span($\mathbf{b}$). In this case, one can use Lemma 1 above, to claim that no information about the subspace $\mathbf{B}$ is revealed (information theoretically).

Suppose for the sake of simplicity, that the adversary that breaches security, actually finds a secret key (rather than merely forging a signature or breaking semantic security). Then, the fact that the random subspace $\mathbf{B}$ is (information theoretically) hidden, implies that with overwhelming probability the secret key $g^{\mathbf{b}'}$ that the adversary outputs is not in $\mathbf{B}$. If indeed $g^{\mathbf{b}'}$ is a valid secret key,

[9] Actually, using such keys, [7] get security under the SXDH assumption. To get security under the linear assumption, they take the public key to be $g^{\mathbf{A}}$ and the secret key to be $g^{\mathbf{B}}$, where $\mathbf{A} \leftarrow \mathbb{Z}_p^{2 \times \ell}$ and $\mathbf{B} \leftarrow \mathbb{Z}_p^{\ell \times 2}$ such that $\mathbf{A} \cdot \mathbf{B} = \mathbf{0}$. This results in a slightly more complicated scheme.

[10] Note that they rely on the DDH assumption, and in addition, in order to verify or decrypt they need to assume the existence of a bilinear map. Thus, they need to rely on the SXDH assumption. However, by taking the public key to be *two* random vectors $\mathbf{A} \leftarrow \mathbb{Z}_p^{2 \times \ell}$ (in the exponent) and the secret key to be two vectors in $\ker(\mathbf{A})$ (in the exponent), they can get security based on the linear assumption over bilinear groups. For this, they use a more general form of Lemma 1.

i.e., $\mathbf{b}' \in \ker(\mathbf{a})$, and $\mathbf{b}'$ is not in the subspace $\mathbf{B}$, then one can use this to break the linear assumption.

This work. In this work, we allow the adversary to both leak and tamper with the secret key. Namely, the adversary can change the secret key from $g^{\mathbf{b}}$ to $T(g^{\mathbf{b}})$. Note that the above proof idea cannot handle tampering, since the update procedure updates a (tampered) secret key by raising it to the power of $\alpha \leftarrow \mathbb{Z}_p^\ell$, i.e., updates $T(g^{\mathbf{b}})$ to $T(g^{\mathbf{b}})^\alpha$. Since the tampering function T is adversarially chosen, we cannot rely on the DDH assumption anymore, and thus cannot use Lemma 1. Indeed, we cannot prove that the scheme of [7] is secure against tampering. Instead, we slightly modify their schemes.

Our schemes. We think of both $g^{\mathbf{a}}$ and $g^{\mathbf{b}}$ as public parameters (or crs), that may be shared among all users, and we assume that these parameters cannot be tampered with. As was mentioned in the introduction, the justification for this assumption is that these parameters are independent of the secret key, and can be thought of as generated in the manufacturing level, and hardwired into the card in a secure way.

Actually, to prove security under the linear assumption (rather than the SXDH assumption), we set the public parameters to be $g^{\mathbf{A}}$ and $g^{\mathbf{B}}$, where $\mathbf{A} \leftarrow \mathbb{Z}_p^{2\times\ell}$ and $\mathbf{B} \leftarrow \mathbb{Z}_p^{\ell\times 2}$ such that $\mathbf{A}\cdot\mathbf{B} = \mathbf{0}$. Then, we let the secret key be $g^{\mathbf{s}}$ where $\mathbf{s} \leftarrow \mathbb{Z}_p^\ell$, and the public key be $e(g,g)^{\mathbf{A}\cdot\mathbf{s}}$, where $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ is a bilinear map. The secret key $g^{\mathbf{s}}$ is updated by multiplying it by $g^{\mathbf{B}\cdot\mathbf{R}}$ where $\mathbf{R} \leftarrow \mathbb{Z}_p^{2\times 2}$. Note that this does not change the public key since

$$e(g,g)^{\mathbf{A}\cdot(\mathbf{s}+\mathbf{B}\cdot\mathbf{R})} = e(g,g)^{\mathbf{A}\cdot\mathbf{s}}.$$

Now, when the adversary tampers with the secret key, and converts it to $T(g^{\mathbf{s}})$, this tampered secret key is updated to be $T(g^{\mathbf{s}})\cdot g^{\mathbf{B}\cdot\mathbf{R}}$, and we can still use the linear assumption to claim that this is computationally indistinguishable from the case that the secret key is updated by adding to it (in the exponent) a random element from a random (independent) subspace $\mathbf{B}' \subseteq \ker(\mathbf{A})$.[11]

We would like to use Lemma 1 to argue that if indeed all the updates are done using $\mathbf{B}'$ (which is a random subspace in $\ker(\mathbf{A})$), then this subspace remains (information theoretically) hidden, even given all the leakages. If we could do so, then we would be in good shape, as before: For the sake of simplicity, suppose that an adversary that breaches security actually outputs a valid secret key (rather than a merely outputting a valid signature or breaking semantic security).[12] Then, this adversary outputs some $g^{\mathbf{s}'}$ such that $e(g,g)^{\mathbf{A}\mathbf{s}'} = e(g,g)^{\mathbf{A}\mathbf{s}}$, and thus $\mathbf{s}' - \mathbf{s} \in \ker(\mathbf{A})$. The fact that $\mathbf{B}'$ is information theoretically hidden implies that with overwhelming probability $\mathbf{s}' - \mathbf{s}$ is not in $\mathrm{span}(\mathbf{B}', \mathbf{B})$, and thus can be used to break the linear assumption.

[11] We defer the details of the actual encryption scheme and signature scheme to Sections 5 and 6, respectively, since they are not of relevance to the discussion here.

[12] Needless to say, this assumption significantly simplifies matters, and the actual proof contains several additional technicality that are hidden from this high level overview.

Unfortunately, when trying to use Lemma 1, we face several technical difficulties. The main problem is the following: Note that we need some affine version of Lemma 1. Indeed, it is quite straightforward to prove that the affine version Lemma 1 is also true (while using Lemma 1 as a black box), assuming the affine element is *independent* of the (hidden) subspace $\mathbf{B}$. Namely, for any $\mathbf{s} \in \mathbb{Z}_p^\ell$

$$(\mathbf{B}, L(\mathbf{s} + \mathbf{B} \cdot \mathbf{r})) \stackrel{\epsilon}{\equiv} (\mathbf{B}, L(\mathbf{u}))$$

where $\mathbf{B} \leftarrow \mathbb{Z}_p^{\ell \times d}$, $\mathbf{r} \leftarrow \mathbb{Z}_p^d$ and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$ (independent of $\mathbf{s}$). However, in our case the adversary, who controls the tampering function T, may cause the affine element $T(g^{\mathbf{s}+\mathbf{BR}})$ to depend in some (arbitrary) way on the subspace $\mathbf{B}$. In this case, we do not know how to prove an affine version of Lemma 1.

Nevertheless, we do succeed in proving an affine variant of Lemma 1 which is secure against tampering. This lemma (Lemma 2), does make use of Lemma 1 (in a black box manner), but requires overcoming several additional technical barriers. We present Lemma 2, which we think of as our main technical lemma, in Section 4, and defer its formal proof to the full version. We note that by updating the secret key using $g^{\mathbf{B}}$, which is part of the public parameters (rather than the secret key), doesn't only allow us to be resilient to tampering, but also improves the leakage bounds of [7]. Known schemes that were proven secure against continual leakage under the linear assumption [7, 10], could resist at most $1/2 - \epsilon$ leakage rate between updates, whereas we can tolerate $1 - \epsilon$ leakage rate between updates.[13]

Roadmap. We define our CTL model in Section 3, followed by a formal statement of our main lemma in Section 4. A formal description of our encryption scheme and our signature scheme can be found in Sections 5 and 6 respectively. Our construction in the model of continual tampering with bounded leakage can be found in Section 7.

3 Our CTL Model

Our CTL model generalizes the continual memory leakage model of [7, 10], to allow the adversary to (continually) tamper with the secret state, as well as (continually) leak. In what follows we define encryption scheme in the CTL model.

Encryption schemes in the CTL *model.* The definition of an encryption scheme in the CTL model is very similar to the definitions given in [7, 10], except for the following difference. In our definition, we partition the key generation algorithm into two parts. The first part is the *public-parameter generation* algorithm, denoted by ppGen. This algorithm takes as input a security parameter 1^k and produces public parameters pp. These public parameters can be thought of as

[13] We note that [7] could tolerate $1 - \epsilon$ leakage rate under the less standard SXDH assumption.

part of the verification key, however we choose to differentiate them from the verification key, since these are independent of the secret key, whereas the verification key does depend on the secret key. The reason for this distinction is that in our results, we do not allow tampering with these public parameters. The justification for this assumption is that these parameters are independent of the secret key, and can be thought of as generated in the manufacturing level, and hardwired into the card in a secure way. We note that the same public parameters can be shared among all users. The second part is the key generation algorithm, denoted by Gen, which takes as input these public parameters pp and the security parameter 1^k, and generates a pair (sk, pk) of secret and public keys.

Formally, an encryption scheme in the CTL model consists of five PPT algorithms:

- The *public-parameter generation algorithm* ppGen takes as input the security parameter 1^k, and outputs public parameters pp. We denote this by $pp \leftarrow \mathsf{ppGen}(1^k)$.
- The *key-generation algorithm* Gen takes as input the security parameter 1^k and public parameters pp, and outputs a pair of secret and public keys (sk, pk). We denote this by $(sk, pk) \leftarrow \mathsf{Gen}(1^k, pp)$.
- *Encryption.* Takes as input the public parameters pp, a public-key pk and a message m in the message space $\mathcal{M}$, and outputs a ciphertext c. We denote this by $c \leftarrow \mathsf{Enc}_{pp,pk}(m)$.
- *Decryption.* Takes as input the public parameters pp, a secret-key sk, and a ciphertext c, and outputs a message m'. We denote this by $m' \leftarrow \mathsf{Dec}_{pp,sk}(c)$.
- *Key-update.* Takes as input the public parameters pp and a secret-key sk, and outputs an "updated" secret-key sk' such that $|sk'| = |sk|$. We denote this by $sk' \leftarrow \mathsf{Update}_{pp}(sk)$.

The correctness requirement is that for all $m \in \mathcal{M}$ and any polynomial t, setting $pp \leftarrow \mathsf{ppGen}(1^k)$, $(sk_0, pk) \leftarrow \mathsf{Gen}(1^k, pp)$ and then computing $sk_i \leftarrow \mathsf{Update}_{pp}(sk_{i-1})$ for $i \in [t]$, we have that for $c \leftarrow \mathsf{Enc}_{pp,pk}(m)$ and $m' \leftarrow \mathsf{Dec}_{pp,sk_t}(c)$, it holds that $m = m'$ with all but negligible probability (where the probability is over all the randomness in the experiment).

We next define semantic security in the CTL model. We use the definition from [7], and augment it by allowing for tampering queries. Formally, the leakage in this model is associated with 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" (relative) memory leakage bound that is enforced between key updates. For the sake of simplicity, we define security for the special case that $\rho_U = 0$. This simplifies the definition and allows for a cleaner exposition. The result of [7], can be used to show that *any* scheme that is secure against continual leakage, can tolerate $O(\log k)$ leakage from each update process, and thus our scheme can tolerate such leakage as well.

Definition 1. *An encryption scheme* $\mathcal{E} = (\mathsf{ppGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$ *is semantically secure in the* CTL *model with leakage rate* $(\rho_G, \rho_U, \rho_M) = (\rho_G, 0, \rho_M)$, *if any* PPT *adversary succeeds in the following game with probability* $1/2+\mathrm{negl}(k)$.

1. Initialize. *The forger specifies a circuit f such that $|f(r)| \leq \rho_G \cdot |r|$ for all r. The challenger chooses "secret randomness" r, generates $(sk_0, pk) = \mathsf{Gen}(1^k, pp; r)$, where pp are the public parameters generated by* ppGen*, sends $(pk, f(r))$ to the adversary, and sets $L := |f(r)|$.*
2. Tampering, leakage and updates. *The adversary makes queries of the following type:*
   - *Update queries* update*. The challenger sets $sk \leftarrow \mathsf{Update}_{pp}(sk)$, and sets $L := 0$.*
   - *Tampering queries (tamper, T), where T is a circuit. The challenger sets $sk := T(sk)$.*
   - *Leakage queries (leak, f), where f is a circuit. If $L + |f(sk)| \leq \rho_M \cdot |sk|$ then the challenger returns $f(sk)$ to the forger, and sets $L := L + |f(sk)|$. Otherwise, the challenger aborts.*
3. Challenge. *The adversary sends two messages m_0, m_1 to the challenger. The challenger flips a coin $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_{pp,pk}(m_b)$, and sends c to the adversary.*
4. Finish. *The adversary outputs a "guess" $b' \in \{0,1\}$.*

The adversary **succeeds** *if $b' = b$.*

Signature schemes in the CTL *model.* The definition of signature scheme in the CTL model is similar to the definition of encryption scheme in the CTL model, except for the following two differences. First, a signature scheme in the CTL model is associated with an additional leakage parameter ρ_S, which captures leakage allowed from the signing process itself. Since our results cannot tolerate leakage from the signing process, throughout this work, we set $\rho_S = 0$. The second difference, which is more subtle, is the following: In the case of signature schemes we bound the number of tampering queries within each time period (however, the number of tampering queries overall is unbounded). The reason is that the adversary has access to a signing oracle, and the signatures themselves (w.r.t. tampered keys) leak some information about the tampered key. We defer the formal definition to the full version.

4 Main Lemma

Intuitively, the main lemma considers the setting where a random vector $\mathbf{s} \leftarrow \mathbb{Z}_p^\ell$ is chosen, and is being continually updated by adding to it either a random element from a small subspace, or a random element from a bigger subspace. In between every two updates the random vector can be tampered with, and partially leaked, many times, as long as the total amount of leakage is bounded. The claim is that it is hard to distinguish between the case where the updates were from a small subspace or from the large subspace, assuming the leakage functions take as input these vectors "in the exponent". More specifically, we consider a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_p^{c \times \ell}$ consisting of c rows, for $2 \leq c < \ell$. The random vector $g^{\mathbf{s}}$ is either updated by adding to it a random element in $\ker(\mathbf{A})$ "in the exponent", or by adding to it a random element in $\mathrm{span}(\mathbf{B})$ "in the

exponent", where $\mathbf{B} \in \mathbb{Z}_p^{\ell \times d}$ is a random d-dimensional subspace of $\ker(\mathbf{A})$. The former is called the *ideal* game, and the latter is called the *real* game, and the claim is that it is computationally hard to distinguish between the two, even given $\mathbf{B}$.

Lemma 2. *For any security parameter k, let $\mathbb{G}$ be a group of order p, where p is a k-bit prime, such that the linear assumption holds in $\mathbb{G}$. Let $\ell, c, d \in \mathbb{N}$ be polynomially bounded parameters such that $c \geq 2$, $4 + c \leq d \leq \ell - c$,[14] and let $t = \mathrm{poly}(k)$. Then, for any* PPT *adversary $\mathcal{A}$,*

$$\mathsf{G}^{\mathcal{A}}_{\mathsf{real}}(p, \ell, c, d, t) \approx \mathsf{G}^{\mathcal{A}}_{\mathsf{ideal}}(p, \ell, c, d, t)$$

as long as each $L_i : \mathbb{G}^\ell \to W_i$ satisfies $|W_i| \leq p^{d-c-4}$, where $\mathsf{G}^{\mathcal{A}}_{\mathsf{real}}(p, \ell, c, d, t)$ and $\mathsf{G}^{\mathcal{A}}_{\mathsf{ideal}}(p, \ell, c, d, t)$ are defined below.

Experiment $\mathsf{G}^{\mathcal{A}}_{\mathsf{real}}(p, \ell, c, d, t)$. In $\mathsf{G}^{\mathcal{A}}_{\mathsf{real}}(p, \ell, c, d, t)$, a challenger $\mathcal{C}$ interacts with the adversary $\mathcal{A}$ as follows:

1. $\mathcal{C}$ chooses a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_p^{c \times \ell}$ and a random vector $\mathbf{s} \leftarrow \mathbb{Z}_p^\ell$. It sets $\mathbf{v}_1 = g^{\mathbf{s}}$, and gives $\mathbf{A}, \mathbf{s}$ to $\mathcal{A}$.
2. $\mathcal{C}$ chooses a random d dimensional subspace $\mathbf{B} \leftarrow \mathbb{Z}_p^{\ell \times d}$ such that $\mathbf{A} \cdot \mathbf{B} = \mathbf{0}$ (i.e., each column in $\mathbf{B}$ is uniformly distributed in $\ker(\mathbf{A})$). For $i = 1$ to t, do:
   (a) $\mathcal{A}$ specifies a leakage queries L_i and a tampering function T_i.
   (b) $\mathcal{C}$ gives $L_i(\mathbf{v}_i)$ to $\mathcal{A}$. In addition he chooses $\mathbf{r}_i \leftarrow \mathbb{Z}_p^d$, sets $\mathbf{b}_i = \mathbf{B} \cdot \mathbf{r}_i$, and sets $\mathbf{v}_{i+1} = T_i(\mathbf{v}_i) \cdot g^{\mathbf{b}_i}$.
3. Finally, $\mathcal{C}$ gives the adversary $\mathcal{A}$ the subspace $\mathbf{B}$, and $\mathcal{A}$ outputs either 0 or 1.

Thus, during this experiment $\mathcal{A}$ gets

$$L_1(\mathbf{v}_1), L_2(\mathbf{v}_2), \ldots, L_t(\mathbf{v}_t),$$

where $\mathbf{v}_1 = g^{\mathbf{s}}$ and $\mathbf{v}_{i+1} = T_i(\mathbf{v}_i) \cdot g^{\mathbf{b}_i}$ for $1 \leq i \leq t-1$.

Experiment $\mathsf{G}^{\mathcal{A}}_{\mathsf{ideal}}(p, \ell, c, d, t)$. This experiment is exactly the same as $\mathsf{G}^{\mathcal{A}}_{\mathsf{real}}(p, \ell, c, d, t)$ with the exception that $\mathbf{b}_i \leftarrow \ker(\mathbf{A})$. For the sake of clarity, we will use the notation $\mathbf{k}_i$ to denote updates in the ideal game (as opposed to $\mathbf{b}_i$ in the real game). That is, in the ideal game, $\mathcal{C}$ sets $\mathbf{v}_{i+1} = T_i(\mathbf{v}_i) \cdot g^{\mathbf{k}_i}$ in Step 2b above.

Thus, during this experiment $\mathcal{A}$ gets

$$L_1(\mathbf{v}_1), L_2(\mathbf{v}_2), \ldots, L_t(\mathbf{v}_t),$$

where $\mathbf{v}_1 = g^{\mathbf{s}}$ and $\mathbf{v}_{i+1} = T_i(\mathbf{v}_i) \cdot g^{\mathbf{k}_i}$ for $1 \leq i \leq t-1$.

We defer the proof of Lemma 2 to the full version.

[14] Think of c as a small constant (such as $c = 2$), think of ℓ as growing polynomially with the security parameter, and think of d as approaching ℓ (such as $d = \ell - c$). Such parameters will optimize our leakage bounds.

5 Encryption Scheme in the CTL Model

In this section, we construct an encryption scheme that is secure in the CTL model. Our encryption scheme $\mathcal{E} = (\mathsf{ppGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$ is defined in Figure 1.

Encryption scheme $\mathcal{E}$

Public-parameter generation. The public-parameter generation algorithm ppGen takes as input the security parameter 1^k, and does as follows:

1. Choose at random $\mathbf{A} \leftarrow \mathbb{Z}_p^{2\times\ell}$ and $\mathbf{B} \leftarrow \mathbb{Z}_p^{\ell\times 2}$ such that $\mathbf{AB} = \mathbf{0}$ (i.e., both of the columns of $\mathbf{B}$ are randomly chosen in $\ker(\mathbf{A})$).

Output $pp = (g^{\mathbf{A}}, g^{\mathbf{B}})$

Key generation. The key generation algorithm Gen, takes as input the security parameter 1^k and public parameters $pp = (g^{\mathbf{A}}, g^{\mathbf{B}})$, and does as follows:

1. Choose a random a vector $g^{\mathbf{s}} \leftarrow \mathbb{G}^{\ell}$ (i.e., $\mathbf{s} \leftarrow \mathbb{Z}_p^{\ell}$).
2. Set $sk = g^{\mathbf{s}}$ and $pk = e(g,g)^{\mathbf{A}\cdot\mathbf{s}}$.

Output (sk, pk) as the secret and public key pair.

Encrypting. The encryption algorithm Enc, takes as input a message $b \in \{0,1\}$, public parameters $pp = (g^{\mathbf{A}}, g^{\mathbf{B}})$, and a public key $pk = e(g,g)^{\mathbf{A}\cdot\mathbf{s}}$, and does as follows:

1. If $b = 0$, choose a random vector $\mathbf{r} \leftarrow \mathbb{Z}_p^2$, and output $(\mathbf{c}_1, c_2) = (g^{\mathbf{r}^T\cdot\mathbf{A}}, e(g,g)^{\mathbf{r}^T\cdot\mathbf{A}\cdot\mathbf{s}})$.
2. If $b = 1$, choose a random vector $\mathbf{r} \leftarrow \mathbb{Z}_p^2$ and a random element $u \leftarrow \mathbb{Z}_p$, and output $(\mathbf{c}_1, c_2) = (g^{\mathbf{r}^T\cdot\mathbf{A}}, e(g,g)^{u})$.

Decrypting. The decryption algorithm Dec, takes as input a ciphertext $\mathbf{c} = (\mathbf{c}_1, c_2)$, public parameters $pp = (g^{\mathbf{A}}, g^{\mathbf{B}})$, and a secret key $sk = g^{\mathbf{s}}$, and does as follows:

1. If $e(\mathbf{c}_1, g^{\mathbf{s}}) = c_2$ then output $b' = 0$. Otherwise, output $b' = 1$.

Updates. The update procedure Update, takes as input public parameters $pp = (g^{\mathbf{A}}, g^{\mathbf{B}})$ and a secret key $sk = g^{\mathbf{s}}$, it chooses $\mathbf{u} \leftarrow \mathbb{Z}_p^2$ and outputs $sk = g^{\mathbf{s}+\mathbf{B}\mathbf{u}}$.

Fig. 1: Encryption scheme in the CTL model.

Theorem 4. *Let $\mathbb{G}$ be a group of prime order p, with a bilinear map $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ and a generator g, and assume that the linear assumption holds in $\mathbb{G}$. Then*

the encryption scheme $\mathcal{E} = (\mathsf{ppGen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$*, described in Figure 1, is semantically secure in the* CTL *model, with leakage rate* $(\rho_G, \rho_U, \rho_M) = (\frac{\ell-13}{\ell}, 0, \frac{\ell-13}{\ell})$.

We defer the proof of this theorem, as well as those of theorems appearing in the subsequent sections, to the full version.

6 Signature Schemes in the CTL Model

In this section, we show how to convert any encryption scheme that is secure in the CTL model into a signature scheme that is secure in the CTL model. Our transformation follows the Katz-Vaikuntanathan paradigm [25] and uses the follows building blocks:

- An encryption scheme $\mathcal{E}_{\mathcal{TL}} = (\mathsf{ppGen}_{\mathcal{TL}}, \mathsf{Gen}_{\mathcal{TL}}, \mathsf{Enc}_{\mathcal{TL}}, \mathsf{Dec}_{\mathcal{TL}}, \mathsf{Update}_{\mathcal{TL}})$ that is semantically secure in the CTL model with leakage rate (ρ_G, ρ_U, ρ_M). We assume the encryption scheme $\mathcal{E}_{\mathcal{TL}}$ has a deterministic predicate T such that $T(pk, sk) = 1$ if and only if sk is a "valid" secret key corresponding to pk (i.e., if sk correctly decrypts ciphertexts encrypted using pk with overwhelming probability).
- An adaptive simulation-sound NIZK proof system $\Pi = (\ell, P, V, S = (S_1, S_2))$ (as defined by Sahai [35]) for the following language L:

$$L = \left\{(m, c, pk', pk) : \exists sk, r \text{ s.t. } c = \mathsf{Enc}_{pk'}(sk; r) \text{ and } T(pk, sk) = 1\right\}.$$

- An (ordinary) semantically secure encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (without leakage or tampering).

Our signature scheme $\mathcal{S} = (\mathsf{ppGen}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Update})$ is defined in Figure 2 and results in the following theorem:

Theorem 5. *The signature scheme* $\mathcal{S} = (\mathsf{ppGen}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Update})$ *described in Figure 2 is existentially unforgeable under adaptive chosen message attacks in the* CTL *model, with leakage rate* $(\rho_G, \rho_U, \rho_S, \rho_M{}')$*, where* $\rho_M{}' = \rho_M - \frac{|vk|}{|sk|}$ *and* $\rho_S = 0$.

In particular, by using the encryption scheme described in Figure 1, we obtain a signature scheme $\mathcal{S} = (\mathsf{ppGen}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Update})$ *that is existentially unforgeable under adaptive chosen message attacks in the* CTL *model, with leakage rate* $(\rho_G, \rho_U, \rho_S, \rho_M) = (\frac{\ell-13}{\ell}, 0, 0, \frac{\ell-15}{\ell})$.

7 Signature Scheme in the Continual Tampering and Bounded Leakage Model

In this section we show how to convert a signature scheme $\mathcal{S}$, that is secure in the bounded leakage model of [1], into a signature scheme that is secure in the

Signature scheme $\mathcal{S}$

Public-parameter generation. The public-parameter generation algorithm ppGen takes as input a security parameter 1^k, and does the following:

1. Sample $pp \leftarrow \mathsf{ppGen}_{\mathcal{TL}}(1^k)$.
2. Sample a public key pk' for the (ordinary) encryption scheme $\mathcal{E}$.
3. sample a random string $\mathsf{crs} \leftarrow \{0,1\}^{\ell(k)}$.

Output (pp, pk', crs) as the public parameters.

Key generation. The key generation algorithm Gen, takes as input the security parameter 1^k and public parameters (pp, pk', crs), and generates $(sk, pk) \leftarrow \mathsf{Gen}_{\mathcal{TL}}(1^k, pp)$. Output sk as the secret key and pk as the verification key.

Signing. Given a message m, public parameters (pp, pk', crs), and a secret key sk, do the following:

1. Choose a random string r and compute $c = \mathsf{Enc}_{pk'}(sk; r)$.
2. Compute a proof π for the statement $(m, c, pk', pk) \in L$ w.r.t. the common random string crs, using (sk, r) as the witness, where

   $$L = \left\{(m, c, pk', pk) : \exists sk, r \text{ s.t. } c = \mathsf{Enc}_{pk'}(sk; r) \text{ and } T(pk, sk) = 1\right\}.$$

   Namely, compute $\pi \leftarrow P_{\mathsf{crs}}((m, c, pk', pk), (sk, r))$.

Output $\sigma = (c, \pi)$ as a signature for m.

Verifying. To verify a signature $\sigma = (c, \pi)$ on a message m with respect to the verification key pk and the public parameters (pp, pk', crs), check whether π is a valid proof of the statement $(m, c, pk', pk) \in L$ with respect to the common random string crs.

Updates. The update procedure Update is identical to that of $\mathsf{Update}_{\mathcal{TL}}$. More specifically, it takes as input the public parameters (pp, pk, crs) and a secret key sk, and updates the secret key by computing $\mathsf{Update}_{\mathcal{TL}}(pp, sk)$.

Fig. 2: Signature scheme in the CTL model.

continual tampering and bounded leakage model[15]. Our construction uses the following primitives as building blocks:

- $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, a signature scheme secure in ρ-bounded leakage model, i.e., $\mathcal{S}$ is existentially unforgeable even if the adversary gets leakage $L(sk)$ of

[15] To enable tampering in the bonded leakage model (where the secret key is never updated), we need to require that the circuit has a self-destruct mechanism i.e., it is possible for the circuit to erase the entire contents of memory. We defer these details to the full version.

his choice such that $| L(sk) | \leq \rho \cdot |sk|$. We assume that $\mathcal{S}$ has the property that $sk \leftarrow \{0,1\}^{g(k)}$ and that there exists a PPT algorithm pkGen such that $vk \leftarrow \mathsf{pkGen}(sk)$ (as is satisfied by [25]).
- A pseudo-random generator $\mathsf{PRG} : \{0,1\}^k \rightarrow \{0,1\}^{g(k)}$.
- A single theorem adaptive "short" simulation sound (SS) NIZK POK [35] $\Pi = (\ell, P, V, S = (S_1, S_2), E)$ for the language $L = \{\psi : \exists s \text{ s.t. } \psi = \mathsf{PRG}(s)\}$, where the length of the proofs are polynomial in the witness length.

We convert $\mathcal{S}$ into $\mathcal{S}' = (\mathsf{ppGen}', \mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$, a signature scheme that is secure against continual tampering and bounded leakage, as depicted in Figure 3. This results in the following theorem:

Theorem 6. *The signature scheme* $\mathcal{S}' = (\mathsf{ppGen}', \mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$*, presented in Figure 3, is existentially unforgeable in the continual tampering and* ρ'*-bounded leakage model, where* $\rho' = \left(\frac{\rho \cdot g(k)}{g(k)+\gamma(k)} - \frac{2k+\gamma(k)}{g(k)+\gamma(k)}\right)$.

References

1. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.
2. Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. Cryptology ePrint Archive, Report 2010/544, 2010. `http://eprint.iacr.org/`.
3. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT*, pages 491–506, 2003.
4. Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, 1997.
5. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT*, pages 37–51, 1997.
6. Victor Boyko. On the security properties of oaep as an all-or-nothing transform. In *CRYPTO*, pages 503–518, 1999.
7. 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 *FOCS*, pages 335–359, 2010.
8. Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.
9. Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.
10. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana Lopez-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, 2010.
11. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.
12. Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In *EUROCRYPT*, pages 301–324, 2001.
13. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.

Signature scheme $\mathcal{S}'$

Public-parameter generation. The public-parameter generation algorithm ppGen' takes as input the security parameter 1^k, and does as follows:

1. Sample a string $\mathsf{crs} \leftarrow \{0,1\}^{\ell(k)}$.

Output $pp' = \mathsf{crs}$

Key generation. The key generation algorithm Gen', takes as input the security parameter 1^k and public parameters $pp' = \mathsf{crs}$, and does as follows:

1. Choose $s \leftarrow \{0,1\}^k$.
2. Set $sk = \mathsf{PRG}(s)$ and choose $vk \leftarrow \mathsf{pkGen}(sk)$.
3. Compute a short SS NIZK POK π for the statement $sk \in L$, w.r.t. crs, where

$$L = \{sk : \exists s' \text{ s.t. } sk = \mathsf{PRG}(s')\}$$

   Namely, $\pi \leftarrow P(sk, s, \mathsf{crs})$.
4. Set $sk' = (sk, \pi)$.

Output (sk', vk) as the secret and verification key pair.

Signing. Given a message m, public parameter $pp' = \mathsf{crs}$, and a secret key $sk' = (sk, \pi)$, do the following:

1. Verify that π is a valid proof for the statement $sk \in L$ with respect to the common random string crs. Namely, check that $V(sk, \pi, \mathsf{crs}) = 1$. If not, self destruct.
2. Compute $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$ (where Sign is the original leakage resilient signing algorithm).

Output σ as a signature for m.

Verifying. To verify a signature σ on a message m w.r.t. the verification key vk, run $\mathsf{Ver}_{vk}(m, \sigma)$ and output whatever it does.

Fig. 3: Signature scheme secure in the continual tampering and bounded leakage model.

14. Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.
15. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.
16. Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting against leakage: the computationally bounded and noisy cases. In *Eurocrypt*, pages 135–156, 2010.
17. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *CHES*, number Generators, pages 251–261, 2001.

18. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.
19. Shafi Goldwasser and Guy Rothblum. How to play mental solitaire under continuous side-channels: A completeness theorem using secure hardware. Manuscript, 2010.
20. Sudhakar Govindavajhala and Andrew W. Appel. Using memory errors to attack a virtual machine. In *IEEE Symposium on Security and Privacy*, pages 154–165, 2003.
21. J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.
22. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.
23. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
24. Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *CRYPTO*, pages 41–58, 2010.
25. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.
26. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.
27. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
28. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.
29. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
30. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of aes. In *CT-RSA*, pages 1–20, 2006.
31. Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.
32. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
33. Josyula R. Rao and Pankaj Rohatgi. Empowering side-channel attacks. Cryptology ePrint Archive, Report 2001/037, 2001. `http://eprint.iacr.org/`.
34. Ronald L. Rivest. All-or-nothing encryption and the package transform. In *FSE*, pages 210–218, 1997.
35. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.