Signature Schemes with Bounded Leakage Resilience

Jonathan Katz[*]

Abstract

A *leakage-resilient* cryptosystem remains secure even if arbitrary information about the secret key (or possibly other internal state information) is leaked to an adversary. We demonstrate the first constructions of leakage-resilient signature schemes that remain secure as long as a *bounded* amount of information, depending on the length n of the secret key, is leaked. We show efficient schemes in the random oracle model that handle leakage of up to $(\frac{1}{2} - \epsilon) \cdot n$ bits of information about the signer's entire internal state. In the standard model, we show an inefficient scheme that can handle leakage of up to $(1 - \epsilon) \cdot n$ bits of information about the secret key, and a one-time signature scheme tolerating arbitrary leakage of $n^{1-\epsilon}$ bits.

1 Introduction

Proofs of security for cryptographic primitives traditionally treat the primitive as a "black box" which an adversary is able to access in a relatively limited fashion. For example, in the usual model for proving security of signature schemes an adversary is given the public key and allowed to request signatures on any messages of its choice, but is unable to get any *other* information about the secret key or any internal randomness or state information used during signature generation.

In real-world implementations of cryptographic primitives, on the other hand, an adversary may be able to recover a significant amount of additional information not captured by standard security models. Examples include information leaked by side-channel cryptanalysis [21, 22], fault attacks [8, 6], or timing attacks [7], or even bits of the secret key itself in case this key is improperly stored or erased [19]. Potentially, schemes can also be attacked when they are implemented using poor random number generation [27] (which can be viewed as giving the adversary additional information on the internal state, beyond that which would be available were the output truly random), or when the same key is used in different contexts (e.g., for decryption and signing).

In the past few years cryptographers have made tremendous progress toward modeling security in the face of such information leakage [24, 34], and in constructing *leakage-resilient* cryptosystems secure even in case such leakage occurs. (There has also been corresponding work on reducing unwanted leakage by, e.g., building tamper-proof hardware; this is not the focus of our work.) Most relevant to the current work is a recent series of results [14, 1, 30, 13, 35, 25] showing cryptosystems that guarantee security even when *arbitrary* information about the secret key is leaked, so long as this information is bounded in some way; we discuss this work, along with other related results, in further detail below. This prior work gives constructions of stream ciphers [14, 30] (and hence stateful symmetric-key encryption and MACs), symmetric-key encryption schemes [13], and public-key encryption schemes [1, 35, 25] achieving various notions of leakage resilience.

The prior work just mentioned concentrates primarily on primitives for ensuring *secrecy*; the case of schemes for guaranteeing *authenticity* has so far been relatively unexplored. Here, we initiate work in this direction and show the first constructions of signature schemes, and hence stateless MACs, that remain secure as long as a bounded amount of information about the key (and other

[*]Dept. of Computer Science, University of Maryland. Work done while visiting IBM. Email: `jkatz@cs.umd.edu`.

internal state information) is leaked. We show schemes that are efficient and can be proven secure in the random oracle model, and also show inefficient, "proof of concept" constructions in the standard model; the levels of leakage resilience achieved in each case are incomparable. We postpone a more thorough discussion until after we define leakage resilience in a bit more detail.

1.1 Modeling Leakage Resilience

We provide a brief overview of the framework in which leakage resilience is defined (specialized to signatures), providing a discussion of previous related work along the way.

At a high level, definitions of leakage resilience take the following form: Begin with a "standard" security notion (e.g., existential unforgeability under adaptive chosen message attacks [17]) and modify this definition by allowing the adversary to (adaptively) specify a series of *leakage functions* $f_1, \ldots$. The adversary, in addition to getting whatever other information is specified by the original security definition, is given the result of applying f_i to the secret key and possibly other internal state of honest party (e.g., the signer). We then require that the adversary's success probability — in the case of signature schemes, the probability with which it can output a forged signature on a previously unsigned message — remain negligible. It should be clear that this is a a general methodology that can be applied to many different primitives.

The exact model is then determined by the restrictions placed on the leakage function(s) f_i. We may distinguish the following possibilities:

Arbitrary or specific information A first issue to be resolved is whether the $\{f_i\}$ are allowed to be arbitrary (polynomial-time computable) functions, or whether they are restricted to be in some more limited class. Early work considered the latter case, for example where adversary is restricted to learning *specific bits* of the secret key [10], or the values on *specific wires* of the circuit implementing the primitive [20]. More recent work [14, 1, 30, 13, 35, 25] allows arbitrary $\{f_i\}$.

Total amount of information bounded or unbounded. Let n denote the length of the secret key. It is clear that if the secret key does not change over time, and the $\{f_i\}$ are allowed to be arbitrary, meaningful security cannot be obtained if the total length of the *leakage* — that is, the outputs of all the $\{f_i\}$ — is n bits or more. (For the case of signatures, the length of the leakage must also be less than the signature length.) This inherent restriction is used in [1, 35, 25].

One can avoid this restriction, and potentially tolerate an unbounded amount of leakage overall, if one is willing to allow the secret key to be updated over time; even in this case, one must somehow limit the amount of leakage between successive key updates. This approach to leakage resilience was considered in [14, 30] in the context of stateful symmetric-key primitives. Updating the secret key is reminiscent of work on *forward-secure* cryptosystems [2, 4, 5, 11], and indeed forward security can be viewed as a limited form of leakage resilience.

One can also avoid imposing a bound on the leakage by restricting the $\{f_i\}$, as discussed next.

Computational min-entropy of the secret key. If the leakage is much shorter than the secret key (as discussed above), then the secret key will have high min-entropy conditioned on the leakage. This setting is considered in [1, 25, 35], and is also enforced on a per-period basis in the work of [14, 30] (i.e., the leakage per time period is required to be shorter than the secret key). More recent work [13, 35] shows schemes that remain secure for certain leakage of arbitrary length, as long as the secret key remains exponentially hard to compute given the leakage (but even if the secret key is fully determined by the leakage in an information-theoretic sense). A drawback of this guarantee is that given some collection of functions $\{f_i\}$ (say, as determined experimentally for some particular set of side-channel attacks) there is no way to tell, in general, whether they satisfy the stated requirement or not. Furthermore, existing results in this direction currently require super-polynomial hardness assumptions.

Inputs to the leakage functions. A final consideration is the allowed inputs of the leakage functions. Work of [14, 30] assumes, following [24], that *only computation leaks information*; this is modeled by letting each f_i take as input only those portions of the secret key that are accessed during the ith phase of the scheme. Halderman et al. [19], however, show that memory contents can be leaked even when they are not being accessed. Motivated (in part) by this result, the schemes of [1, 13, 35, 25] allow the $\{f_i\}$ to take the entire secret key as input at all times.

For the specific primitives considered in [14, 1, 30, 13, 35, 25], the secret key sk is the only internal state maintained by the party holding the secret key, and so allowing the $\{f_i\}$ to depend on sk is (almost) the most general choice.[1] We observe that for other primitives, and in particular for signature schemes with randomized signing, there is additional state information that is kept secret from the adversary in the traditional security definition. The strongest definition of leakage resilience is thus obtained by allowing the $\{f_i\}$ to depend on *all* the state information used by the honest signer during the course of the experiment.

All these variants may be meaningful depending on the particular attacks one is trying to model. Memory attacks [19, 1], which probe long-term secret information during a time when computation is *not* taking place, can be faithfully modeled by allowing the leakage functions to take only sk as input. On the other hand, side-channel attacks that collect information while computation is occurring might be more accurately captured by allowing the leakage functions to take as input only those portions of the internal state that are actively be accessed.

1.2 Our Results

With the preceding discussion in mind, we can now describe our results in further detail. In all cases, we allow the leakage functions to be *arbitrary* as long as the total leakage is *bounded* as some function of the secret key length n; recall that such a restriction on the leakage is essential if the secret key is unchanging, as it is in all our schemes.

1. We show a class of efficient, leakage-resilient signature schemes in the random oracle model. For any constant $\epsilon > 0$, these schemes tolerate leakage of $(\frac{1}{2} - \epsilon) \cdot n$ bits that may depend on *the entire state* of the signer throughout the course of the experiment.

2. We also show an inefficient construction of a leakage-resilient signature scheme in the standard model. In this case we can tolerate more leakage — up to $(1 - \epsilon) \cdot n$ bits — but restrict the leakage functions to depend on the secret key (and the randomness used to generate it) only. (Thus, here we offer protection against the "memory attacks" of [1].)

3. Finally, we show a leakage-resilient *one-time* signature scheme in the standard model. (The same idea can also be extended to give a t-time signature scheme.) This scheme is more efficient than the previous one and, as in the case of our first scheme, tolerates leakage that may depend on the entire state of the signer throughout the experiment. (In fact, here signing is deterministic.) This scheme tolerates leakage of $n^{1-\epsilon}$ bits, asymptotically less than the previous two schemes.

[1] More generally, one could also allow the $\{f_i\}$ to depend on the *randomness* used to generate the (public and) secret key(s); this possibility is mentioned in [25, Section 8.2]. (For the specific schemes considered in [14, 1, 30, 13, 35, 25], however, this makes no substantive difference.) One could also allow the $\{f_i\}$ to be applied to the state of *other* honest parties besides the holder of the secret key, e.g., the sender in a public-key encryption scheme. This issue has not yet been addressed in any work of which we are aware.

Our constructions all rely on the same basic idea. Roughly, we consider signature schemes with the following properties:

- A given public key pk corresponds to a set S_{pk} of *exponentially many* secret keys. Furthermore, given (sk, pk) with $sk \in S_{pk}$ it remains hard to compute any other $sk' \in S_{pk}$.
- The "real" secret key sk used by the signer should be computationally indistinguishable from a random secret key in S_{pk}, even for an adversary who observes signatures on arbitrary messages of its choice.
- A signature forgery can be used to compute a secret key in S_{pk}.

To prove, informally, that any such signature scheme is leakage resilient, we show how to use an adversary $\mathcal{A}$ attacking the scheme to find distinct $sk, sk' \in S_{pk}$ given (sk, pk) (in violation of the assumed hardness of doing so). Given (sk, pk), we simply run $\mathcal{A}$ on input pk and respond to its signing queries using the given key sk. Leakage queries can also be answered without any problem using sk. When the adversary forges a signature, we extract $sk' \in S_{pk}$; it remains only to show that $sk' \neq sk$ with high probability. Let $k = \log |S_{pk}|$ be the initial min-entropy of sk conditioned on pk. As long as the leakage is bounded by $\alpha \cdot k$ bits for some $\alpha < 1$, with high probability the min-entropy of sk, conditioned on $\mathcal{A}$'s entire view, is still at least 2. But then $sk' \neq sk$ with probability at least 1/2. This concludes the outline of the proof. We remark, however, that getting the details to work is a bit more subtle for our constructions in the random oracle model.

Some existing signature schemes in the random oracle model already satisfy the requirements stated above. In particular, these include schemes constructed using the Fiat-Shamir transform [15] applied to a witness-indistinguishable Σ-protocol where there are an *exponential* number of witnesses corresponding to a given statement. Concrete examples include the signature schemes of Okamoto [28] (extending the Schnorr [33] and Guillou-Quisquater [18] schemes) based on the discrete logarithm or RSA assumptions, as well as the signature scheme of Fischlin and Fischlin [16] (extending the Ong-Schnorr [29] scheme) based on the hardness of factoring.

We are not aware of any existing signature scheme in the standard model that meets our requirements. We construct one as follows. Let H be a universal one-way hash function (UOWHF) [26] mapping n-bit inputs to (ϵn)-bit outputs. The secret key of the signature scheme is $x \in \{0,1\}^n$, and the public key is $(y = H(x), pk, r)$ where pk is a public key for a CPA-secure public-key encryption scheme, and r is a common reference string for an unbounded simulation-sound NIZK proof system [32, 12]. A signature on a message m consists of an encryption $C \leftarrow \mathsf{Enc}_{pk}(m\|x)$ of both m and x, along with a proof π that C is an encryption of $m\|x'$ with $H(x') = y$. Observe that, with high probability over choice of x, there are exponentially many pre-images of $y = H(x)$ and hence exponentially many valid secret keys; furthermore, finding another such secret key $sk' \neq sk$ requires finding a collision in H.

Our leakage-resilient one-time signature scheme is constructed using a similar idea, applied to the Lamport one-time signature scheme [23]. That is, the secret key is $\{(x_{i,0}, x_{i,1})\}_{i=1}^k$ and the public key is $\{(y_{i,0}, y_{i,1})\}_{i=1}^k$ where k is the message length and $y_{i,b} = H(x_{i,b})$ for H a UOWHF as above. Once again, there are exponentially many secret keys associated with any public key and finding any two such keys yields a collision in H.

2 Definitions

We provide a formal definition of leakage resilience for signature schemes. We denote the security parameter by k, and let PPT stand for "probabilistic polynomial time".

Definition 1 A *signature scheme* is a tuple of PPT algorithms (Gen, Sign, Vrfy) such that:

- Gen is a randomized algorithm that takes as input 1^k and outputs (pk, sk), where pk is the public key and sk is the secret key.
- Sign is a (possibly) randomized algorithm that takes as input the secret key sk, the public key pk, and a message m, and outputs a signature σ. We denote this by $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$, leaving the public key implicit.[2]
- Vrfy is a deterministic algorithm that takes as input a public key pk, a message m, and a purported signature σ. It outputs a bit b indicating acceptance or rejection, and we write this as $b := \mathsf{Vrfy}_{pk}(m, \sigma)$.

It is required that for all k, all (pk, sk) output by $\mathsf{Gen}(1^k)$, and all messages m in the message space, we have $\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m)) = 1$. $\diamondsuit$

Our definition of leakage resilience is the standard notion of existential unforgeability under adaptive chosen-message attacks [17], except that we additionally allow the adversary to specify arbitrary leakage functions $\{f_i\}$ and obtain the value of the functions applied to the secret key (and possibly other state information).

Definition 2 Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme, and let λ be a function. Given an adversary $\mathcal{A}$, define the following experiment parameterized by k:

1. Set $\mathsf{state} := \emptyset$.
2. Choose $r \leftarrow \{0,1\}^*$ and compute $(pk, sk) := \mathsf{Gen}(1^k; r)$. Set $\mathsf{state} := \mathsf{state} \cup \{r\}$.
3. Run $\mathcal{A}(1^k, pk)$. The adversary may then adaptively access a *signing oracle* $\mathsf{Sign}_{sk}(\cdot)$ and a *leakage oracle* $\mathsf{Leak}_{\mathsf{state}}(\cdot)$ that have the following functionality:
   - In response to the ith query $\mathsf{Sign}_{sk}(m_i)$, this oracle chooses random $r_i \leftarrow \{0,1\}^*$, computes $\sigma_i := \mathsf{Sign}_{sk}(m_i; r_i)$, and returns σ_i to $\mathcal{A}$. It also sets $\mathsf{state} := \mathsf{state} \cup \{r_i\}$.
   - In response to the ith query $\mathsf{Leak}_{\mathsf{state}}(f_i)$ (where f_i is specified as a circuit), this oracle gives $f_i(\mathsf{state})$ to $\mathcal{A}$. (To make the definition meaningful in the random oracle model, the $\{f_i\}$ are allowed to be oracle circuits that depend on the random oracle H.)
4. At some point, $\mathcal{A}$ outputs (m, σ).

We say $\mathcal{A}$ *succeeds* if (1) $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$; (2) m was not previously queried to the $\mathsf{Sign}_{sk}(\cdot)$ oracle; and (3) the total output length of all the f_i is at most $\lambda(|sk|)$. We denote the probability of this event by $\Pr[\mathsf{Succ}^{\lambda\text{-leakage}}_{\mathcal{A},\Pi}(k)]$. We say Π is *λ-leakage resilient* if $\Pr[\mathsf{Succ}^{\lambda\text{-leakage}}_{\mathcal{A},\Pi}(k)]$ is negligible for every PPT adversary $\mathcal{A}$.

If, in the above experiment, state is not updated after each signing query (and so always contains only the randomness r used to generate the secret key), we denote the probability of success by $\Pr[\mathsf{Succ}^{\lambda\text{-memory}}_{\mathcal{A},\Pi}(k)]$ and say Π is *secure against a λ-memory attacks* if $\Pr[\mathsf{Succ}^{\lambda\text{-memory}}_{\mathcal{A},\Pi}(k)]$ is negligible for every PPT adversary $\mathcal{A}$. $\diamondsuit$

Other variations of the definition are possible: state could include only sk (rather than the random coins r used to generate it), or could include only the most recently used random coins r_i. The definition above reflects the most general notions of leakage/memory attacks.

[2]Usually, one can assume without loss of generality that the secret key includes whatever portion of the public key is needed to sign. Since we measure leakage as a function of the secret-key length in this work, however, we seek to minimize the size of the secret key.

3 Leakage-Resilient Signatures in the Random Oracle Model

We exemplify the approach described earlier using (a variant of) the Okamoto-Schnorr signature scheme [33, 28], thus basing security on the discrete logarithm assumption. The same ideas can also be applied to obtain schemes based on the RSA [18, 28] or factoring [29, 16] assumptions (but see the remark at the end of this section).

Let $\mathcal{G}$ be a group generation algorithm that on input 1^k outputs $(\mathbb{G}, q)$ where q is a k-bit prime and $\mathbb{G}$ is a cyclic group of order q. We assume group operations in $\mathbb{G}$ are efficient, and that elements of $\mathbb{G}$ can be sampled "obliviously": i.e., there is an efficient algorithm $\mathsf{samp}_{\mathbb{G}}$ generating random elements of $\mathbb{G}$ with the property that, given $g \in \mathbb{G}$, one can sample uniformly from the set of coins ω for which $g := \mathsf{samp}_{\mathbb{G}}(\omega)$.[3] (We leave $\mathsf{samp}_{\mathbb{G}}$ implicit from now on.)

Our signature scheme is parameterized by an integer ℓ, and is defined as follows:

Key generation: Compute $(\mathbb{G}, q) \leftarrow \mathbb{G}(1^k)$ and choose random $g_1, \ldots, g_\ell \leftarrow \mathbb{G}$ and random $x_1, \ldots, x_\ell \leftarrow \mathbb{Z}_q$. Set $h := \prod_i g_i^{x_i}$. The public key is $(\mathbb{G}, q, g_1, \ldots, g_\ell, h)$ and the secret key is $(x_1, \ldots, x_\ell)$. We also assume a random oracle $H : \{0,1\}^* \to \mathbb{Z}_q$.

Signing: Signatures are perfectly witness-indistinguishable proofs of knowledge of a representation of h with respect to the basis $(g_1, \ldots, g_\ell)$, turned into signatures using the Fiat-Shamir transformation. In detail: to sign message m using secret key $(x_1, \ldots, x_\ell)$ and public key $(\mathbb{G}, q, g_1, \ldots, g_\ell, h)$, the signer chooses random $r_1, \ldots, r_\ell \leftarrow \mathbb{Z}_q$ and computes $A := \prod_i g_i^{r_i}$. It then sets $c := H(A, m)$, and outputs the signature $(A, cx_1 + r_1, \ldots, cx_\ell + r_\ell)$.

Verification: Given a signature $(A, \alpha_1, \ldots, \alpha_\ell)$ on the message m with respect to the public key $(\mathbb{G}, q, g_1, \ldots, g_\ell, h)$, compute $c := H(A, m)$ and output 1 iff $\prod_i g_i^{\alpha_i} \stackrel{?}{=} h^c \cdot A$.

Before proving security, we formally state the discrete logarithm assumption as well as an equivalent assumption we will use in our proof.

Definition 3 Let $\mathcal{G}$ be as above. We say the *discrete logarithm problem is hard for $\mathcal{G}$* if the following is negligible for all PPT algorithms $\mathcal{A}$:

$$\Pr\left[(\mathbb{G}, q) \leftarrow \mathcal{G}(1^k); g, h \leftarrow \mathbb{G}; x \leftarrow \mathcal{A}(\mathbb{G}, q, g, h, \omega) : g^x = h\right],$$

where ω denotes the randomness used to generate $\mathbb{G}, q, g, h$. (Note that by our assumption on $\mathsf{samp}_{\mathbb{G}}$, the coins used to generate g, h are extraneous.) $\diamondsuit$

We stress that the above definition requires hardness to hold even against adversaries given the randomness used to generate the problem instance. (For concrete $\mathcal{G}$ used in practice, this additional randomness does not seem to make the problem any easier.)

Definition 4 Let $\mathcal{G}$ be as above. We say the *ℓ-representation problem is hard for $\mathcal{G}$* if the following is negligible for all PPT algorithms $\mathcal{A}$:

$$\Pr\left[\begin{array}{c}(\mathbb{G}, q) \leftarrow \mathcal{G}(1^k); g_1, \ldots, g_\ell \leftarrow \mathbb{G}; \\ (x_1, \ldots, x_\ell), (x'_1, \ldots, x'_\ell) \leftarrow \mathcal{A}(\mathbb{G}, q, g_1, \ldots, g_\ell, \omega)\end{array} : \prod_i g_i^{x_i} = \prod_i g_i^{x'_i} \bigwedge \vec{x} \neq \vec{x}'\right],$$

where ω denotes the randomness used to generate $\mathbb{G}, q, g_1, \ldots, g_\ell$. $\diamondsuit$

Hardness of the discrete logarithm problem for $\mathcal{G}$ implies hardness of the ℓ-representation problem [9, 3] (for any polynomial ℓ). This implication carries over to our setting (where the randomness used to generate the instance is given to the adversary) given our assumption on $\mathsf{samp}_{\mathbb{G}}$.

[3]This property holds for most concrete examples $\mathcal{G}$ used in practice.

Theorem 1 *If the discrete logarithm problem is hard for $\mathcal{G}$, then for any $\epsilon > 0$ the signature scheme above is $\left(\frac{1}{2} - \frac{1}{2\ell} - \epsilon\right) \cdot n$-leakage resilient, where n denotes the length of the secret key.*

Thus, for any desired $1 > \epsilon' > 0$, setting $\ell > 1/\epsilon'$ (and taking $\epsilon = \epsilon'/2$) gives a scheme that is $(1 - \epsilon') \cdot n$-leakage resilient.

Proof Let $\mathcal{A}$ be a probabilistic polynomial-time adversary attacking the scheme in the sense of Definition 2. We let q_H be a bound on the total number of hash queries asked throughout the entire experiment; such queries can be made directly by $\mathcal{A}$ and can also occur in the course of answering signing or leakage queries. Denote the success probability of $\mathcal{A}$ by $\delta = \delta(k)$.

We make a number of assumptions about $\mathcal{A}$ without loss of generality. First, we assume that if $\mathcal{A}$ outputs $(m, \sigma = (A, \alpha_1, \ldots, \alpha_\ell))$ then (1) $\mathcal{A}$ at some point queried $H(A, m)$ and (2) $\mathcal{A}$ never requested a signature on m. Second, for any leakage query $\mathsf{Leak}_{\mathsf{state}}(f_i)$ we assume f_i makes the same number of H-oracle calls regardless of the value of state (this can always be ensured by adding dummy queries, as needed).

We construct a probabilistic polynomial-time algorithm $\mathcal{B}$ solving the ℓ-representation problem. Algorithm $\mathcal{B}$ proceeds as follows: on input $(\mathbb{G}, q, g_1, \ldots, g_\ell, \omega)$, it chooses random $x_1, \ldots, x_\ell$ and computes $h := \prod_i g_i^{x_i}$. It gives the public key $pk = (\mathbb{G}, q, g_1, \ldots, g_\ell, h)$ to $\mathcal{A}$ and runs the entire experiment of Definition 2, simulating the random oracle for $\mathcal{A}$. Note that $\mathcal{B}$ can run the entire experiment easily since it knows a legitimate secret key $(x_1, \ldots, x_\ell)$ corresponding to pk.

When $\mathcal{A}$ terminates, $\mathcal{B}$ examines $\mathcal{A}$'s output $(m, \sigma = (A, \alpha_1, \ldots, \alpha_\ell))$; we call the execution of $\mathcal{A}$ to this point the *first run of* $\mathcal{A}$. If $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$, then $\mathcal{B}$ rewinds to the point in the experiment where the hash query $H(A, m) = c$ was first made; note that this may occur either as a result of a direct H-query made by $\mathcal{A}$, or during the course of answering a Leak query. $\mathcal{B}$ then chooses a fresh random value c' for the result of $H(A, m)$, and re-runs the experiment from that point; we will refer to the execution of $\mathcal{A}$ from this point to termination as the *second run of* $\mathcal{A}$. During the second run of $\mathcal{A}$, signing queries are answered using fresh randomness but H-oracle queries are answered consistently with the first run of $\mathcal{A}$. (I.e., if a hash query is made during the second run of $\mathcal{A}$ that was also made during the first run of $\mathcal{A}$, the same answer chosen during the first run is used during the second run. Any new hash queries are answered using fresh randomness.) We say $\mathcal{B}$ *succeeds* if, in this second run, $\mathcal{A}$ terminates with output $(m, \sigma' = (A, \alpha'_1, \ldots, \alpha'_\ell)$ where σ' is also a valid signature on m.

If $\mathcal{B}$ succeeds and also $c' \neq c$, then $\mathcal{B}$ computes (in the standard way [28]) values $(x'_1, \ldots, x'_\ell)$ such that $\prod_i g_i^{x'_i} = h$. By definition, $\mathcal{B}$ solves the representation problem if $(x'_1, \ldots, x'_\ell) \neq (x_1, \ldots, x_\ell)$. The following two claims complete the proof of the theorem.

Claim 1 $\Pr[\mathcal{B} \text{ succeeds}] \geq \delta^2/q_H$.

Proof Consider all possible states during the execution of $\mathcal{A}$ in the experiment of Definition 2 where a new hash query is made, and for any such state i let h_i denote the hash query made at that state. If an execution of $\mathcal{A}$ terminates with a valid forgery $(m, \sigma = (A, \alpha_1, \ldots, \alpha_\ell))$, say that state i is *associated with that forgery* if $h_i = (m, A)$.

For any state i where a new hash query is made, let a_i be the probability that this state is reached in the experiment of Definition 2, where this probability is over the entire specification of the random oracle H (*except* for the value of $H(h_i)$, which does not affect the probability since the query h_i is being made for the first time in state i) as well as the randomness of $\mathcal{A}$, the randomness used to generate the public key, and the randomness used to answer any signing queries made up to that point. For the same state i, let b_i be the probability that, starting from state i, the execution of $\mathcal{A}$ terminates with a successful forgery associated with i. This probability is over the value of

$H(h_i)$ and the randomness used to answer any signing queries made after this point. Since every successful forgery is associated with a unique state of $\mathcal{A}$, we have $\sum_i a_i \cdot b_i = \delta$, the overall success probability of $\mathcal{A}$. Furthermore, we have $\sum_i a_i = \mathbf{E}[\text{number of hash queries made by } \mathcal{A}] \leq q_h$.

By construction of $\mathcal{B}$, we have $\Pr[\mathcal{B} \text{ succeeds}] = \sum_i a_i \cdot (b_i)^2$. Using Jensen's inequality:

$$\begin{aligned} \sum_i a_i \cdot (b_i)^2 &\geq \sum_i a_i \cdot \left(\frac{\sum_i a_i \cdot b_i}{\sum_i a_i}\right)^2 \\ &\geq \frac{\delta^2}{\sum_i a_i} \geq \delta^2/q_H\,, \end{aligned}$$

completing the proof of the claim. □

Claim 2 *The probability that $\mathcal{B}$ solves the representation problem is at least*

$$\frac{1}{2} \cdot \left(\Pr[\mathcal{B} \text{ succeeds}] - 1/q - q_H/q^{2\epsilon\ell}\right).$$

Proof When $\mathcal{B}$ succeeds, there are two bad events that can prevent $\mathcal{B}$ from solving the representation problem. First, it may be the case that $c' = c$; this happens with probability $1/q$. Second, it may be the case that the extracted representation $\vec{x}' = (x'_1, \ldots, x'_\ell)$ is equal to the original representation $\vec{x} = (x_1, \ldots, x_\ell)$. We show that except with probability at most $q_H/q^{\epsilon\ell}$, the min-entropy of $\vec{x}$ conditioned on the two views of $\mathcal{A}$ (in the two runs of $\mathcal{A}$) is at least 2; given this, the probability that $\vec{x}' \neq \vec{x}$ is at least 1/2 and the claim follows.

Let $\lambda = \left(\frac{1}{2} - \frac{1}{2\ell} - \epsilon\right) \cdot \ell \cdot \log q$, an upper bound on the number of leaked bits in each run of $\mathcal{A}$. The public key pk constrains $\vec{x}$ to lie in an $(\ell-1)$-dimensional vector space, and it is well-known [28] that signature queries do not further constrain $\vec{x}$. Thus, the min-entropy of $\vec{x}$ conditioned on the public key and the results of the signature queries is $(\ell-1) \cdot \log q$ bits. The two views of $\mathcal{A}$ in its two runs contain only the following additional information about $\vec{x}$: at most $2 \cdot \lambda$ bits from the leakage functions (i.e., λ bits in each view), and $\log q_H$ bits indicating the relevant state associated with the first forgery (cf. the proof of Claim 1).[4] A standard argument thus shows that the probability that the conditional min-entropy of $\vec{x}$ is less than 2 is at most

$$\left(q_H \cdot 2^{2\lambda}\right) \cdot 2^{-(\ell-1)\cdot\log q} \leq q_H \cdot q^{-2\epsilon\ell}.$$

The claim follows. □

Taking the two claims together, we see that if $\mathcal{A}$ succeeds with probability δ then $\mathcal{B}$ solves the representation problem with probability at least

$$\frac{1}{2} \cdot \left(\delta^2/q_H - 1/q - q_H/q^{2\epsilon\ell}\right).$$

Since q_H is polynomial, $\epsilon > 0$ is a constant, and $1/q$ is negligible, the above is non-negligible whenever δ is. This completes the proof of the theorem. ■

[4]Note that this information may not be evident from $\mathcal{A}$'s view. For example, consider the leakage query f_1 defined as follows: "if $x_1 = 0$ then query $H(A, m)$ (otherwise do nothing); in any case, return the first bit of x_2." If $\mathcal{A}$ (later queries $H(A, m)$ and) eventually outputs a forgery $(A, \ldots)$ on m, then the state associated with this forgery depends on $\vec{x}$ and cannot be determined solely from the view of $\mathcal{A}$.

It is an interesting open question to improve the tightness of the security reduction. Although we expect it to be possible to show an algorithm $\mathcal{B}'$ solving the representation problem with probability $\mathcal{O}\left(\delta - 1/q^{2\epsilon\ell}\right)$ (given $\mathcal{A}$ with success probability δ), our efforts to prove such a bound have failed.

Remark: Instantiating this approach using factoring-based assumptions. As noted at the beginning of this section, the same approach as above can also be applied to obtain leakage-resilient schemes based on the RSA [18, 28] or factoring [29, 16] assumptions. In these cases, however, we must either (1) restrict the leakage functions to apply only to those values used by the signer *after* the key generation phase (i.e., the secret key and all state variables used to sign, but not the random coins used to generate the secret/public key), or (2) generate the modulus N via oblivious sampling, leading to a significant loss in efficiency.

4 A Signature Scheme Secure Against Memory Attacks

We construct a signature scheme *in the standard model* that is secure against memory attacks, following the intuition described in Section 1.2. Let (Gen_H, H) be a UOWHF [26] mapping n-bit inputs to $\epsilon \cdot n$-bit outputs for some $n = \mathsf{poly}(k)$; we assume that finding second pre-images is hard even given the randomness used to generate the key. (Several constructions of UOWHFs believed to have this property are known.) Let $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure, dense[5] public-key encryption scheme; and let $(\ell, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2)$ be an unbounded simulation-sound NIZK proof system [12] for the following language L:

$$L = \{(s, y, pk, m, C) : \exists x, \omega \text{ s.t. } C = \mathsf{Enc}_{pk}(x; \omega) \text{ and } H_s(x) = y\}.$$

The signature scheme is defined as follows:

Key generation: Choose random $x \leftarrow \{0,1\}^n$ and compute $s \leftarrow \mathsf{Gen}_H(1^k)$. Obliviously sample a public key pk for the encryption scheme, and choose a random string $r \leftarrow \{0,1\}^{\ell(k)}$. The public key is $(s, y := H_s(x), pk, r)$ and the secret key is x.

Signing: To sign message m using secret key x and public key (s, y, pk, r), first choose random ω and compute $C \leftarrow \mathsf{Enc}_{pk}(x; \omega)$. Then compute a proof $\pi \leftarrow \mathcal{P}_r((s, y, pk, m, C), (x, \omega))$; i.e., a proof that $(s, y, pk, m, C) \in L$ using witness (x, ω). The signature is (C, π).

Verification: Given a signature (C, π) on the message m with respect to the public key (s, y, pk, r), output 1 iff $\mathcal{V}_r((s, y, pk, m, C), \pi) = 1$.

Theorem 2 *Under the stated assumptions on the components, the signature scheme above is secure against* $(\frac{1}{2} - 2\epsilon) \cdot n$*-memory attacks.*

Proof (Sketch) Let Π denote the scheme given above, and let $\mathcal{A}$ be a PPT adversary with $\delta = \delta(k) \overset{\text{def}}{=} \Pr[\mathsf{Succ}^{\lambda\text{-memory}}_{\mathcal{A},\Pi}(k)]$. We consider a sequence of experiments, and let $\Pr_i[\cdot]$ denote the probability of an event in experiment i. The experiments are defined as follows:

Experiment 0: This is the experiment of Definition 2. Given the public key (s, y, pk, r) defined by the experiment, let Succ denote the event that $\mathcal{A}$ outputs $(m, (C, \pi))$ where $\mathcal{V}_r((s, y, pk, m, C), \pi) = 1$ and m was never queried to its signing oracle. By assumption, we have $\Pr_0[\mathsf{Succ}] = \delta$.

[5]This means it is possible to sample a valid public key without knowing the corresponding secret key.

Experiment 1: We introduce the following differences with respect to the preceding experiment: when setting up the public key, we now generate r by computing $(r, \tau) \leftarrow \mathcal{S}_1(1^k)$. Furthermore, signing queries are now answered as follows: to sign m, generate C as before but compute π as $\pi \leftarrow \mathcal{S}_2((s, y, pk, m, C), \tau)$.

Define event Succ as in the preceding experiment. By the (adaptive) zero-knowledge property of $(\ell, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2)$, it must be the case that $|\Pr_1[\mathsf{Succ}] - \Pr_0[\mathsf{Succ}]|$ is negligible.

Experiment 2: We modify the preceding experiment in the following way: to answer a signing query for a message m, compute $C \leftarrow \mathsf{Enc}_{pk}(0^n)$ (and then compute π as in Experiment 1). Defining Succ as in the preceding two experiments, CPA-security of the encryption scheme implies that $|\Pr_2[\mathsf{Succ}] - \Pr_1[\mathsf{Succ}]|$ is negligible.

Experiment 3: We now change the way the public key is generated. Namely, instead of obliviously sampling the encryption public key pk we compute it as $(pk, sk) \leftarrow \mathsf{Gen}_E(1^k)$. Note that this is only a syntactic change and so $\Pr_3[\mathsf{Succ}] = \Pr_2[\mathsf{Succ}]$. (This assumes perfect oblivious sampling; if an obliviously generated public key and a legitimately generated public key are only computationally indistinguishable, then the probability of Succ can be affected by a negligible amount.)

Given the public key (s, y, pk, r) defined by the experiment, let Ext be the event that $\mathcal{A}$ outputs $(m, (C, \pi))$ such that $H_s(\mathsf{Dec}_{sk}(C)) = y$. Unbounded simulation soundness of the NIZK proof system implies that $|\Pr_3[\mathsf{Ext}] - \Pr_3[\mathsf{Succ}]|$ is negligible. (Note that the message m is included in any statement in the language L, and so if $\mathcal{A}$ did not request a signature on m then it was never given a fake proof of the statement (s, y, pk, m, C).)

To complete the proof, we show that $\Pr_3[\mathsf{Ext}]$ must be negligible. Specifically, consider the following adversary $\mathcal{B}$ finding a second-preimage in the UOWHF: $\mathcal{B}$ chooses random $x \leftarrow \{0,1\}^n$ and is given key s (along with the randomness used to generate s). $\mathcal{B}$ then runs Experiment 3 with $\mathcal{A}$. We stress that $\mathcal{B}$ can do this with no trouble since $\mathcal{B}$ knows a legitimate secret key x for the public key that it gives to $\mathcal{A}$; thus, in particular, $\mathcal{B}$ can answer any leakage queries asked by $\mathcal{A}$. (Recall further that here we are only concerned with *memory attacks* where the leakage functions can be applied to the secret key, and the randomness used to generate it, but not to any auxiliary state used during signing.) If event Ext occurs when $\mathcal{A}$ terminates, then $\mathcal{B}$ recovers a value $x' \stackrel{\text{def}}{=} \mathsf{Dec}_{sk}(C)$ for which $H_s(x') = y = H_s(x)$. We know argue that $x' \neq x$ with high probability.

The only information about x revealed to $\mathcal{A}$ in Experiment 3 comes from y and the leakage queries asked by $\mathcal{A}$, and these total at most $\epsilon \cdot n + (1 - 2\epsilon) \cdot n = (1 - \epsilon) \cdot n$ bits. The probability that the conditional min-entropy of x is less than 2 is there at most $2^{(1-\epsilon)n} \cdot 2^{-n} = 2^{-\epsilon n}$. When the conditional min-entropy of x is at least 2, we have $x' \neq x$ with probability at least $\frac{1}{2}$. Thus, the probability that $\mathcal{B}$ extracts an x' as above with $x' \neq x$ is at least

$$\frac{1}{2} \cdot \left(\Pr_3[\mathsf{Ext}] - 2^{-\epsilon n}\right).$$

We thus see that if $\delta = \Pr_0[\mathsf{Succ}]$ is non-negligible then $\mathcal{B}$ violates the security of the UOWHF with non-negligible probability, a contradiction. ■

5 A Leakage-Resilient One-Time Signature Scheme

We briefly describe a construction of a leakage-resilient one-time signature scheme in the standard model. The advantages of this construction with respect to the scheme described in the previous section is two-fold: first, it is more efficient; second, it is leakage-resilient rather than only secure against memory attacks. The disadvantage, besides the fact that it is only a one-time signature scheme, is that the leakage that can be tolerated is much lower.

Let (Gen_H, H) be a UOWHF mapping k^c-bit inputs to k-bit outputs for some $c > 0$. (As before, we assume H is secure even given the randomness used to generate the key.) Our signature scheme is a variant on Lamport's scheme [23], using H as the one-way function. (This same construction also gives a *strong* one-time signature scheme based on any one-way function [26, 31].)

Key generation: Choose random $x_{i,0}, x_{i,1} \leftarrow \{0,1\}^{k^c}$ for $i = 1, \ldots, k$. Also generate $s \leftarrow \mathsf{Gen}_H(1^k)$. Compute $y_{i,b} := H_s(x_{i,b})$ for $i \in \{1, \ldots, k\}$ and $b \in \{0,1\}$. The public key is $(s, \{y_{i,b}\})$ and the secret key is $\{x_{i,b}\}$.

Signing: The signature on a k-bit message $m = m_1, \ldots, m_k$ consists of the k values $x_{1,m_1}, \ldots, x_{k,m_k}$.

Verification: Given a signature $x_1, \ldots, x_k$ on the k-bit message $m = m_1, \ldots, m_k$ with respect to the public key $(s, \{y_{i,b}\})$, output 1 iff $y_{i,m_i} \overset{?}{=} H_s(x_i)$ for all i.

Theorem 3 *If (Gen_H, H) is a UOWHF, then the above scheme is a $n^{(c-1)/(c+1)}$-leakage resilient one-time signature scheme, where $n = 2k^{c+1}$ denotes the length of the secret key.*

Thus, for any desired $1 > \epsilon > 0$, setting $c = 2\epsilon^{-1} - 1$ gives a one-time signature scheme that is $n^{1-\epsilon}$-leakage resilient.

We omit the proof, which follows the same ideas as those used in the proof of Theorem 2. Note that the scheme can be extended using cover-free families, in the natural way, to yield a leakage-resilient t-time signature scheme for any polynomial t.

Acknowledgments

I am grateful to Yael Tauman Kalai and Vinod Vaikuntanathan for stimulating discussions about leakage resilience, and explications of [1, 13, 35].

References

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

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

[3] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In *Advances in Cryptology — Crypto '94*, volume 839 of *LNCS*, pages 216–233. Springer, 1994.

[4] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *Advances in Cryptology — Crypto '99*, volume 1666 of *LNCS*, pages 431–448. Springer, 1999.

[5] M. Bellare and B. S. Yee. Forward-security in private-key cryptography. In *Cryptographers' Track — RSA 2003*, volume 2612 of *LNCS*, pages 1–18. Springer, 2003.

[6] E. Biham, Y. Carmeli, and A. Shamir. Bug attacks. In *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 221–240. Springer, 2008.

[7] D. Boneh and D. Brumley. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[8] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology — Eurocrypt '97*, volume 1233 of *LNCS*, pages 37–51. Springer, 1997.

[9] S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, 1993.

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

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

[12] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology — Crypto 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, 2001.

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

[14] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 293–302. IEEE Computer Society Press, 2008. Full version available at `http://eprint.iacr.org/2008/240`.

[15] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[16] M. Fischlin and R. Fischlin. The representation problem based on factoring. In *Cryptographers' Track — RSA 2002*, volume 2271 of *LNCS*, pages 96–113. Springer, 2002.

[17] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[18] L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology — Crypto '88*, volume 403 of *LNCS*, pages 216–231. Springer, 1990.

[19] A. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino, A. Feldman, J. Applebaum, and E. Felten. Lest we remember: Cold boot attacks on encryption keys. In *Proc. 17th USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.

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

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

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

[23] L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.

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

[25] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. Available at `http://eprint.iacr.org/2009/105`.

[26] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43. ACM Press, 1989.

[27] P. Q. Nguyen and I. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15(3):151–176, 2002.

[28] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology — Crypto '92*, volume 740 of *LNCS*, pages 31–53. Springer, 1993.

[29] H. Ong and C.-P. Schnorr. Fast signature generation with a Fiat-Shamir-like scheme. In *Advances in Cryptology — Eurocrypt '90*, volume 473 of *LNCS*, pages 432–440. Springer, 1990.

[30] K. Pietrzak. A leakage-resilient mode of operation. In *Advances in Cryptology — Eurocrypt 2009*, LNCS, pages ???–??? Springer, 2009.

[31] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394. ACM Press, 1990.

[32] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 543–553. IEEE Computer Society Press, 1999.

[33] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology — Crypto '89*, volume 435 of *LNCS*, pages 239–252. Springer, 1990.

[34] F.-X. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology — Eurocrypt 2009*, LNCS, pages ???–??? Springer, 2009.

[35] Y. Tauman and V. Vaikuntanathan. Personal communication from the second author, March 2009.