Leakage-Resilient Signatures

Sebastian Faust[1,⋆], Eike Kiltz[2,⋆⋆], Krzysztof Pietrzak[2], and Guy N. Rothblum[3]

[1] K.U. Leuven ESAT-COSIC/IBBT, Belgium
[2] CWI, Amsterdam, The Netherlands
[3] MIT, Boston, USA

Abstract. The strongest standard security notion for digital signature schemes is unforgeability under chosen message attacks. In practice, however, this notion can be insufficient due to "side-channel attacks" which exploit leakage of information about the secret internal state. In this work we put forward the notion of "leakage-resilient signatures," which strengthens the standard security notion by giving the adversary the additional power to learn a bounded amount of *arbitrary information* about the secret state that was accessed during *every signature generation*. This notion naturally implies security against all side-channel attacks as long as the amount of information leaked on each invocation is bounded and "only computation leaks information."

The main result of this paper is a construction which gives a (tree-based, stateful) leakage-resilient signature scheme based on any 3-time signature scheme. The amount of information that our scheme can safely leak *per signature generation* is 1/3 of the information the underlying 3-time signature scheme can leak *in total*. Signature schemes that remain secure even if a bounded total amount of information is leaked were recently constructed, hence instantiating our construction with these schemes gives the first constructions of provably secure leakage-resilient signature schemes.

The above construction assumes that the signing algorithm can sample truly random bits, and thus an implementation would need some special hardware (randomness gates). Simply generating this randomness using a leakage-resilient stream-cipher will in general not work. Our second contribution is a sound general principle to replace uniform random bits in any leakage-resilient construction with pseudorandom ones: run two leakage-resilient stream-ciphers (with independent keys) in parallel and then apply a two-source extractor to their outputs.

1 Introduction

Traditionally, provable security treats cryptographic algorithms as black-boxes. An adversary may have access to inputs and outputs, but the computation within

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-11799-2_36

⋆ Supported in part by Microsoft Research through its PhD Scholarship Programme and FWO grant G.0225.07. This work has partly be done while visiting CWI.

⋆⋆ Supported by the research program Sentinels (http://www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

D. Micciancio (Ed.): TCC 2010, LNCS 5978, pp. 343–360, 2010.

the box stays secret. In particular, the standard security notion of digital signatures is existential unforgeability under chosen message attacks [16] (UF-CMA), where one requires that an adversary cannot forge a valid signature even when given access to a signing oracle.

Unfortunately, this traditional security model often does not match reality where an adversary can attack the algorithm's *implementation* with more powerful attacks. An important example in this context are side-channel attacks, which provide an adversary with a partial view on the inner secret state (e.g., a secret signing key) of an algorithm's execution due to physical leakage during computation. In the last two decades a vast number of ingenious side-channel attacks have been invented and used to break implementations of schemes which were provably secure in the traditional model. Examples of side-channels include information derived from running-time [20], electromagnetic radiation [30,14], power consumption [21], and many more (see, e.g., [31,27]).

1.1 Leakage-Resilient Cryptography

Classical research in side-channel attacks sometimes resembles a cat-and-mouse game. New side-channel attacks are discovered, and then heuristic countermeasures are proposed to prevent the specific new attack. This yields countermeasures that are tailored specifically for the class of attacks they intend to defeat. Not very surprisingly, these countermeasures are often later found to be vulnerable to new attacks. This state of affairs is fundamentally different from the design principles of "modern cryptography," where one usually requires that the system is secure against all adversaries from some well defined resource bounded class[1] and for a broad and well-defined attack scenario. (E.g., existential unforgeability for signature schemes or IND-CCA2 security for encryption.)

As this situation is clearly not very satisfying, in an influential paper Micali and Reyzin [24] suggest a framework for adapting the methodology of modern cryptography to the scenario of side-channel attacks.

A FORMAL SECURITY DEFINITION. Inspired by the framework of Micali-Reyzin and Maurer's bounded storage model (and the subsequent bounded-retrieval model), in [10] the notion of *leakage-resilience* was proposed.[2] A cryptographic primitive (or protocol) is said to be leakage-resilient, if it is secure in the traditional (black-box) sense but now the adversary may additionally obtain *arbitrary side-channel information* (also called *leakage*) during the execution of the

[1] In complexity based cryptography one usually bounds the running time. Other bounds that often are used include the size of the memory an adversary can use or the number of queries the adversary can make to some oracle.

[2] The primary contribution of [10] was not proposing a new model, their model combined ideas that were explicit and implicit in prior work. Rather, the primary contribution was actually constructing a primitive (a stream-cipher) and proving it secure in this model.

security experiment. The side-channel information given to the adversary only has to satisfy the following two restrictions

LR1 (bounded leakage): the *amount* of leakage *in each invocation* is bounded (but overall can be arbitrary large).

LR2 (only computation leaks information): the internal state that is not accessed during an invocation ("passive state") does not leak.

At a technical level this is modeled by considering adversaries that, when attacking the primitive, additionally to the regular input specify a *leakage function* f with bounded range $\{0,1\}^\lambda$ and then (besides the regular output) also obtain $\Lambda = f(s^+, r)$, where s^+ denotes the part of the internal secret state that has been accessed during this invocation ("active state") and r are the internal coin tosses that were made by the cryptosystem during the invocation.

Motivation of the leakage restrictions. It is clear that one has to restrict the class of leakage functions, as if we would allow the identity function $f(s) = s$ (where s is the cryptographic algorithm's internal state) , no security whatsoever can be achieved.

In this work we focus on leakage functions that are restricted in terms of their output length. This is a natural resource bound, and allows to model a rich class of side-channel attacks (e.g. timing or hamming-weight attacks, which exploit only a polylogarithmic amount of information on each invocation. This is much smaller than the constant-fraction leakage for which we can still prove security in this work.) We remark that we could use a more relaxed restriction than a bound on the leakage function,[3] but we will stick to *bounded leakage* (LR1) which is more intuitive and simpler to work with.

Bounded leakage alone might not be a *sufficiently* strong restriction, and we use a further restriction on the leakage function, which still seems to allow a rich and very natural family of side-channel attacks. Following [10], we use LR2 ("only computation leaks information"), originally put forward as one of the axioms of "physically observable cryptography" by Micali and Reyzin [24]. The original axiom requires that if a primitive with secret internal state s is invoked, then on this particular invocation, only the part $s^+ \subseteq s$ of the memory leaks that was accessed during this invocation.

Let us stress that the restrictions LR1 and LR2 on the leakage are sufficient, but not necessary to imply security of a leakage-resilient primitive. For example, in a so called "cold-boot attack" [17], the adversary learns (at some point in time) a random subset of the bits of the *entire* secret state, even when no computation is going on. Such an attack clearly does not satisfy the "only computation leaks information" restriction, but nonetheless, leakage-resilience does imply security against cold-boot attacks for any primitive which satisfies some

[3] In particular, we can consider the class $\mathcal{F}$ of leakage functions such that the degradation of the HILL-pseudoentropy of the internal state S due to leakage of $f(S)$ (where $f \in \mathcal{F}$) is sufficiently bounded.

natural additional properties on how the memory is accessed. In particular, these properties are satisfied by our construction and [10,29].[4]

1.2 Leakage-Resilient Signatures

Previous work has shown how to build stream-ciphers that are provably resistant to continual leakage in the standard model [10,29]. In this paper we construct a leakage-resilient public-key primitive in the plain model, a signature-scheme.

Digital signatures are a central cryptographic primitive and are widely implemented on computational devices that are especially vulnerable to side-channel attacks (such as smart cards). Starting with the seminal work by Kocher [20], there have been a great number of theoretical and practical side-channel attacks on signature schemes (e.g., [20,21,32,13]).

Security Definition. The standard notion for secure signatures schemes is unforgeability under adaptive chosen-message attacks [16]. Here one requires that an adversary cannot forge a signature of any message m, even when given access to a signing oracle. We strengthen this notion by giving the adversary access to a more powerful oracle, which not only outputs signatures for chosen messages, but as an additional input takes a leakage function $f : \{0,1\}^* \to \{0,1\}^\lambda$ and outputs $f(s^+, r)$ where s^+ is the state that has been accessed during computation of the signature and (if the scheme is probabilistic) r is the randomness that was sampled. Note that if we want the signature scheme to sign a large number of messages (i.e., more than the state length), then this security definition inherently requires the signature scheme to update its internal state. We call signature schemes which are secure in the above sense UF-CMLA (unforgeable under chosen message/leakage attacks) or simply leakage resilient. We also define a notion called UF-CMTLA (unforgeability under chosen message *total* leakage attacks), which is defined similarly to UF-CMLA but is significantly weaker, as here the total amount of leakage (and not the leakage per invocation) is bounded.

Overview of our construction. Our construction of leakage resilient signature schemes is done in two steps. First, we give a number of instantiations of 3-time UF-CMTLA signature schemes offering different trade-offs. Then, we present a generic tree-based transformation from any UF-CMTLA secure 3-time signature scheme (i.e., a signature scheme that can securely sign up to 3 messages) to a UF-CMLA signature scheme.

From UF-CMTLA to UF-CMLA Security. Following the construction of Naor and Yung [26] and the ideas of Lamport [22] and Merkle [23], we propose a simple tree-based leakage-resilient signature scheme SIG^* that is constructed from any leakage resilient 3-time signature scheme SIG. The scheme we propose

[4] One requirement is that ultimately the entire initial secret state is touched. (The only setting we are aware of where the entire state will *not* be touched, are the the tokens used in the construction of one-time programs [15].) The second requirement is that the memory access is "oblivious", in the sense that which parts of the memory are accessed does not depend on the secret state.

strongly resembles the construction of a forward-secure signature scheme [3] from [4], but let us stress that leakage-resilience and forward-security are orthogonal concepts. In particular, our construction is *not* forward-secure, but could be made so in a straight forward way, at the cost of having a more complicated description.

For any a-priori fixed $d \in \mathbb{N}$, our construction can sign up to $2^{d+1}-2$ messages and one can think of the (stateful) signing algorithm as traversing the $2^{d+1}-1$ nodes of a binary tree of depth d in a depth-first manner. Suppose the signing algorithm of SIG^* wants to sign the i-th message m and its state points to the i-th node $\tilde{w}$ in a depth-first traversal of the tree. It first computes a fresh public/secret-key pair $(pk_{\tilde{w}}, sk_{\tilde{w}})$ of SIG for this node. Next, the signature (σ, Γ) for m is computed, where σ is a signature on m according to the 3-time signature scheme SIG using the secret key $sk_{\tilde{w}}$ of the current node $\tilde{w}$, and Γ contains a signature path from the root of the tree to the node $\tilde{w}$: for each node w on the path it contains a signature on pk_w using the secret key $sk_{\mathsf{par}(w)}$, where $\mathsf{par}(w)$ denotes the parent of w in the tree. The public-key of SIG^* is the public-key associated to the root node and verification of a signature of SIG^* is done by verifying all the 3-time signatures on the path from $\tilde{w}$ to the root.

The crucial observation that will allow us to prove leakage-resilience of our construction, is that for each node w in the tree, the secret key sk_w associated to this node is only accessed a constant number of times (at most three times). The security we prove roughly states that if SIG is a UF-CMTLA secure 3-time signature scheme which is secure even after leaking a total of λ_{total} bits, then SIG^* is a UF-CMLA secure signature scheme that can tolerate $\lambda = \lambda_{\text{total}}/3$ bits of leakage per signature query. The loss in security is a factor of q.

Instantiations UF-CMTLA secure 3-time signature schemes. It is not hard to see that every signature scheme looses at most an exponential factor $2^{\lambda_{\text{total}}}$ in security (compared to the standard UF-CMA security) when λ_{total} bits about the secret key are leaked (as the UF-CMA adversary can simply guess the leakage, and a random guess will be correct with probability $2^{-\lambda_{\text{total}}}$). Recently, much better constructions have been proposed. Alwen, Dodis, and Wichs [2] show that the Okamoto-Schnorr signature-scheme [28,33] remains secure even if almost $n/2$ bits (where n is the length of the secret key) of information about the secret-key are leaked. Instantiating our construction with Okamoto-Schnorr signatures thus gives a leakage resilient signature scheme which can leak a constant fraction (almost 1/6) of the accessed state on each invocation. Due to the Fiat-Shamir heuristic used in the Okamoto-Schnorr signature scheme, this scheme can only be proven secure in the random-oracle model. Recently, Katz and Vaikuntanathan [19] showed how to construct signature schemes in the standard model (and under standard assumptions) which can tolerate leakage of as much as $\lambda_{\text{total}} = n - n^{\epsilon}$ bits ($\epsilon > 0$). With this construction we get a leakage resilient signature scheme in the standard model. Unfortunately it is not practical due to the use of general NIZK proofs.

In the same paper [19], Katz et al. also construct an efficient *one-time* signature scheme that tolerates leakage of $\lambda_{\text{total}} = (1/4 - \epsilon)n$ bits (for any $\epsilon > 0$).

This scheme is easily generalized to a (stateful) 3-time signature schemes where one can leak $\lambda_{\text{total}} = (1/12 - \epsilon)n$ bits.[5] This construction fits well into our general transformation, yielding a UF-CMLA secure scheme where one can leak $\lambda_{\text{total}} = (1/36 - \epsilon)n$ bits (here n is the size of the accessed state on each invocation). As the construction only assumes universal one-way hash functions (UOWHF), we get that it is secure in the standard model under the minimal [26] assumption that one-way functions exist.

1.3 Replacing Randomness in Leakage-Resilient Primitives

In the construction of SIG^* we silently assumed that the device could sample uniformly random bits to be used in the key-generation and signing steps of the underlying scheme SIG. This , however, would require special hardware for generating random bits (such as noise generating gates). In the non-leakage setting one can avoid the necessity for such special hardware by using pseudorandomness (generated by a stream-cipher) instead of truly random bits.

Unfortunately, in the leakage-setting the simple analogous idea of replacing the random bits with the output of a *leakage-resilient* stream-cipher (as defined in [10]) does not work (at least we do not know how to prove it). The reason is that an output block of a leakage-resilient stream-cipher is only guaranteed to have high HILL-pseudoentropy *when given the leakage* that was generated while computing this block.

A sound approach to replace uniform random bits in *any* leakage-resilient construction while provably preserving leakage-resilience is as follows: run two leakage-resilient stream ciphers with independent keys in parallel and feed their output to a two-source extractor. For lack of space, this can be found in the full version [11]. Intuitively, the reason is that now the outputs X, X' of the two stream ciphers are indistinguishable from having high min-entropy (given the leakage), and thus applying a two source extractor ext gives a (indistinguishable from) uniform $Y = \mathsf{ext}(X, X')$ which then can be used in the signature scheme.

While we do not know how to prove in general the security of the simpler approach of using a single leakage-resilient stream cipher to generate the random bits, in some special cases this simpler approach does go through. For example:

– If the scheme (for which we want to replace the uniform random bits) already can be proven leakage-resilient assuming only that the random bits have high min-entropy (as opposed to being uniform), this is e.g. the case for the (generalized) Okamoto signature scheme from [2].
– The output of the particular leakage-resilient stream-cipher from [10] can always be used directly. Informally, the reason is that here (unlike e.g. in [29]) the final output already was generated by applying an extractor.

[5] They propose a general transformation to t-time schemes using cover free sets which can leak $\lambda_{\text{total}} = \Omega(n/t^2)$ bits (which for $t = 3$ is $\Omega(n)$). We note however, that (while this leakage bound is worse than ours) their scheme enjoys the advantage of being stateless, whereas ours is stateful.

1.4 Related Work

A body of prior work has considered countermeasures against different classes of side-channel attacks. Most works consider security against some particular attack, like “template attacks” [34]. Below we mention some work on “provable security” in the context of side-channel attacks, where only the class of leakage functions is restricted, but not the adversaries ability to exploit the leakage.

Ishai *et al.* [18] show how to securely implement *any* (efficiently computable) function even when the attacker can probe a bounded number of wires in the implementation. This result has been recently extendend [12] to allow leakage functions that get as input the values carried by all the wires in the circuit, as long as the output of the leakage functions is short, and the leakage functions is from some low complexity class like AC_0.

Micali and Reyzin [24] proposed the influential theoretical framework of “physically observable cryptography” to model side-channel attacks. In particular, they explicitly state and motivate the “only computation leaks information” axiom used in leakage-resilient cryptography [10,29].

Several recent works [1,25,19,7] propose (stateless) constructions which are secure against so called “memory attacks”. This means that they remain secure even after a bounded total amount of information has leaked (this is sufficient against attacks like “cold-boot” attacks [17], but not for most other side-channel attacks which leak on each invocation). Unlike in leakage-resilient cryptography, here the leakage functions need not obey the only “computation leaks information” restriction. Akavia et. al [1] and Naor and Segev [25] construct symmetric/public-key encryption schemes that are secure in this model. Katz et al [19] and Alwen et al [2] construct digital signatures in this setting (see the discussion above). The “bounded retrieval model” (BRM) [5,8,9,2] is an extension of “memory attacks” where the key is made artificially huge and thus the tolerated leakage can also be made arbitrary large (but still a priory bounded by the key size). The difficulty in this model (as compared to memory attacks), is that in the BRM model the efficiency of a scheme must only depend on some security parameter, but not on the size of the (potentially huge) secret key. Dodis et al. [7,6] consider the case where the range of $f(\cdot)$ is not necessarily bounded, but instead one only requires that it is (exponentially) hard to recover sk from $f(sk)$.

2 Preliminaries

Notion. If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. For $n \in \mathbb{N}$, we write $[n]$ as shorthand for $\{1, \ldots, n\}$. If S is a set then $s \overset{\$}{\leftarrow} S$ denotes the operation of picking an element s of S uniformly at random. With PPT we denote probabilistic polynomial time.

Algorithms. We write $y \leftarrow \mathcal{A}(x)$ to indicate that $\mathcal{A}$ is an algorithm which runs on input x and outputs y. If $\mathcal{A}$ is probabilistic, $y \overset{\$}{\leftarrow} \mathcal{A}(x)$ denotes running the algorithms using fresh randomness.

To model *stateful algorithms* we will in particular consider algorithms with a special input/output syntax. We split the input into three disjoint syntactic parts: a query x, the state s, and (in case the algorithm is probabilistic) randomness r. Similarly, the output is split into the output y and the new state s'. We write $(y, s') \leftarrow \mathcal{B}(x, s, r)$ to make this explicit. Here one can think of the query x as being chosen (or at least known) to the adversary. The state s and s' is the secret internal state of the primitive before and after execution of the algorithm on input x, respectively.

If we consider the execution $(y, s') \leftarrow \mathcal{B}(x, s, r)$ of an algorithm, we can split the state in two parts $s = s^+ \cup s^-$. The *active state*, s^+, denotes the part that is accessed by $\mathcal{B}$ in order to compute y and update its state.[6] The *passive state*, $s^- = s \setminus s^+$, is the part of the state that is not accessed (i.e., read and/or overwritten) during the current execution. We use the notation

$$(y, s') \overset{s^+}{\hookleftarrow} \mathcal{B}(x, s, r) \ .$$

to make explicit that s^+ is the active state of the execution of $\mathcal{B}$ with inputs x, s, r. This is illustrated in Figure 1. Note that the passive state s^- is completely contained in s', i.e., state information that is never accessed is contained entirely in the next state s'.

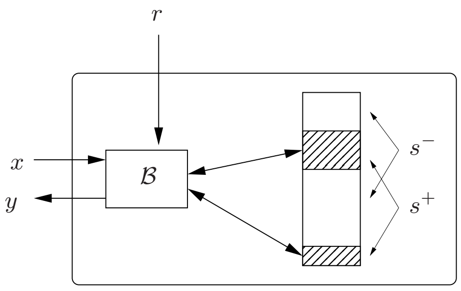

Fig. 1. Illustration of the execution of a stateful algorithm $(y, s') \overset{s^+}{\hookleftarrow} \mathcal{B}(x, s, r)$. The secret state s splits into the active state s^+ (that is accessed during the execution of $\mathcal{B}$) and the passive state s^-.

3 Leakage Resilient Signatures

3.1 Standard Signatures

A (stateful) digital signature scheme $\mathsf{SIG} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ consists of three PPT algorithms. The key generation algorithm Kg generates a secret signing key sk and a public verification key pk. The signing algorithm Sign get as input the

[6] For this to be well defined, we really need that $\mathcal{B}$ is given as an algorithm, e.g. in pseudocode, and not just as a function.

signing key sk and a message m and returns a signature and a new state sk' which replaces the old signing key. The deterministic verification algorithm Vfy inputs the verification key and returns 1 (accept) or 0 (reject). We demand the usual correctness properties.

We recall the definition for unforgeability against chosen-message attacks (UF-CMA) for stateful signatures. To an adversary $\mathcal{F}$ and a signature scheme $\mathsf{SIG} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ we assign the following experiment.

```
Experiment $\mathbf{Exp}_{\mathsf{SIG}}^{\text{uf-cma}}(\mathcal{F}, k)$
    $(pk, sk_0) \xleftarrow{\$} \mathsf{Kg}(1^k)$ ; $i \leftarrow 1$
    $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{F}^{\mathcal{O}_{sk_{i-1}}}(pk)$
    If $\mathsf{Vfy}(pk, m^*, \sigma^*) = 1$ and $m^* \notin \{m_1, \dots m_i\}$
        then return 1 else return 0.

Oracle $\mathcal{O}_{sk_{i-1}}(m_i)$
    $(\sigma_i, sk_i) \xleftarrow{\$} \mathsf{Sign}(sk_{i-1}, m_i)$
    Return $\sigma_i$ and
    set $i \leftarrow i + 1$
```

We remark that for the special case where the signature scheme is stateless (i.e., $sk_{i+1} = sk_i$), we can consider a simpler experiment where the signing oracle $\mathcal{O}_{sk_i}(\cdot)$ is replaced by $\mathsf{Sign}(sk, \cdot)$. With $\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cma}}(\mathcal{F}, k)$ we denote the probability that the above experiment returns 1. Forger $\mathcal{F}$ (t, q, ϵ)-breaks the UF-CMA security of SIG if $\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cma}}(\mathcal{F}, k) \geq \epsilon$, its running time is bounded by $t = t(k)$, and it makes at most $q = q(k)$ signing queries. We call SIG *UF-CMA secure* (or simply *secure*) if no forger can (t, q, ϵ)-break the UF-CMA security of SIG for polynomial t and q and non-negligible ϵ.

3.2 Leakage Resilient Signatures

We now define the notion of unforgeability against chosen-message/leakage attacks (UF-CMLA) for stateful signatures. This extends the UF-CMA security notion as now the adversary can learn λ bits of leakage with every signature query. With the ith signature query, the adversary can adaptively choose any leakage function f_i (described by a circuit[7]) with range $\{0, 1\}^\lambda$ and then learns the output Λ_i of f_i which as input gets everything the signing algorithm gets, that is the active state S_{i-1}^+ and the random coins r_i. To an adversary $\mathcal{F}$ and a signature scheme $\mathsf{SIG} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$ we assign the following experiment.

```
Experiment $\mathbf{Exp}_{\mathsf{SIG}}^{\text{uf-cmla}}(\mathcal{F}, k, \lambda)$
    $(PK, SK_0) \xleftarrow{\$} \mathsf{Kg}(1^k)$ ; $i \leftarrow 1$
    $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{F}^{\mathcal{O}_{SK_{i-1}}}(PK)$
    If $\mathsf{Vfy}(PK, m^*, \sigma^*) = 1$ and
        $m^* \notin \{m_1, \dots m_i\}$
        then return 1 else return 0.

Oracle $\mathcal{O}_{SK_{i-1}}(m_i, f_i)$
    Sample fresh randomness $r_i$
    $(\sigma_i, SK_i) \overset{SK_{i-1}^+}{\hookleftarrow} \mathsf{Sign}(SK_{i-1}, m_i, r_i)$
    $\Lambda_i \leftarrow f_i(SK_{i-1}^+, r_i)$
    if $|\Lambda_i| \neq \lambda$ then $\Lambda_i \leftarrow 0^\lambda$
    Return $(\sigma_i, \Lambda_i)$ and set $i \leftarrow i + 1$
```

[7] We could also model the f_i's as Turing machines, but then we would have to require that the output length is independent of the input, as otherwise information could be encoded in the output length itself.

With $\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cmla}}(\mathcal{F}, k, \lambda)$ we denote the probability that the above experiment returns 1. Forger $\mathcal{F}$ $(t, q, \epsilon, \lambda)$-breaks the UF-CMLA security of SIG if its running time is bounded by $t = t(k)$, it makes at most $q = q(k)$ signing queries and $\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cmla}}(\mathcal{F}, k, \lambda) \geq \epsilon(k)$. We call SIG *UF-CMLA secure with* λ *leakage* (or simply λ-*leakage resilient*) if no forger can $(t, q, \epsilon, \lambda)$-break the UF-CMLA security of SIG for polynomial t and q and non-negligible ϵ.

3.3 Signatures with Bounded Total Leakage

In the previous section we defined signatures that remain secure even if λ bits leak on each invocation. We will construct such signatures using as building block signature schemes that can only sign a constant number (we will need 3) of messages, and are unforgeable assuming that a *total* of λ_{total} bits are leaked (including from the randomness r_0 that was used at key-generation). Following [19], we augment the standard UF-CMA experiment with an oracle $\mathcal{O}_{\text{leak}}$ which the adversary can use to learn up to λ_{total} arbitrary bits about the randomness used in the entire key generation and signing process. This oracle will use a random variable $\mathtt{state}$ that contains all the random coins used by the signature scheme so far and a counter λ_{cnt} to keep track how much has already been leaked. Note that we do not explicitly give the leakage functions access to the key sk_i, as those can be efficiently computed given $r_0 \in \mathtt{state}$.

```
Experiment $\mathbf{Exp}_{\mathsf{SIG}}^{\text{uf-cmtla}}(\mathcal{F}, k, \lambda_{\text{total}})$
    $(pk, sk_0) \overset{r_0}{\leftarrow} \mathsf{Kg}(1^k)$;  $i \leftarrow 1$;  $\lambda_{\text{cnt}} \leftarrow 0$;  $\mathtt{state} \leftarrow r_0$
    $(m^*, \sigma^*) \overset{\$}{\leftarrow} \mathcal{F}^{\mathcal{O}_{sk_{i-1}}, \mathcal{O}_{\text{leak}}}(pk)$
    If $\mathsf{Vfy}(pk, m^*, \sigma^*) = 1$ and $m^* \notin \{m_1, \ldots m_i\}$
        then return 1 else return 0.
```

```
Oracle $\mathcal{O}_{sk_{i-1}}(m_i)$
    Sample fresh randomness $r_i$
    $\mathtt{state} \leftarrow \mathtt{state} \cup r_i$
    $(\sigma_i, sk_i) \leftarrow \mathsf{Sign}(sk_{i-1}, m_i, r_i)$
    Return $\sigma_i$ and set $i \leftarrow i + 1$
```

```
Oracle $\mathcal{O}_{\text{leak}}(f)$
    $\Lambda \leftarrow f(\mathtt{state})$
    If $\lambda_{\text{cnt}} + |\Lambda| > \lambda_{\text{total}}$ Return $\perp$
    $\lambda_{\text{cnt}} \leftarrow \lambda_{\text{cnt}} + |\Lambda|$
    Return $\Lambda$
```

With $\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cmtla}}(\mathcal{F}, k, \lambda_{\text{total}})$ we denote the probability that the above experiment returns 1. Forger $\mathcal{F}$ $(t, d, \epsilon, \lambda_{\text{total}})$-breaks the UF-CMTLA security of SIG if its running time is bounded by $t = t(k)$, it makes at most $d = d(k)$ signing queries and $\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cmtla}}(\mathcal{F}, k, \lambda_{\text{total}}) \geq \epsilon(k)$. We call SIG *UF-CMTLA secure with* λ_{total} *leakage* if no forger can $(t, d, \epsilon, \lambda_{\text{total}})$-break the UF-CMTLA security of SIG for polynomial t and non-negligible ϵ.

4 Construction of Leakage Resilient Signature Schemes

We first discuss three constructions of UF-CMTLA secure 3-time signature schemes. We then prove our main result which shows how to get a leakage-resilient signature scheme from any UF-CMTLA 3-time signatures scheme using a tree based construction.

4.1 Signatures with Bounded Leakage Resilience

Generic construction with exponential loss. We first present a simple lemma showing that *every* d-time UF-CMA secure signature scheme is also a d-time UF-CMTLA secure signature scheme, where the security loss is exponential in λ_{total}.

Lemma 1. *For any security parameter k, $t = t(k)$, $\epsilon = \epsilon(k)$, $d = d(k)$, and λ_{total}, if SIG is (t, d, ϵ) UF-CMA secure, then SIG is $(t', d, 2^{\lambda_{\text{total}}}\epsilon, \lambda_{\text{total}})$ UF-CMTLA secure where $t' \approx t$.*

Proof. For contradiction, assume there exists an adversary $\mathcal{F}_{\lambda_{\text{total}}}$ who breaks the $(t', d, 2^{\lambda_{\text{total}}}\epsilon, \lambda_{\text{total}})$ UF-CMTLA security. We will show how to construct an adversary $\mathcal{F}$ which on input a public-key pk breaks the (t, d, ϵ) UF-CMA security of SIG in a chosen message attack. $\mathcal{F}^{\mathcal{O}_{sk_{i-1}}}(pk)$ simply runs $\mathcal{F}_{\lambda_{\text{total}}}^{\mathcal{O}_{sk_{i-1}}, \mathcal{O}_{\text{leak}}}(pk)$, where it randomly guesses the output of the leakage oracle $\mathcal{O}_{\text{leak}}$. As $\mathcal{O}_{\text{leak}}$ outputs at most λ_{total} bits, $\mathcal{F}$ will guess all the leakage correctly with probability $2^{-\lambda_{\text{total}}}$. Conditioned on $\mathcal{F}$ guessing correctly, $\mathcal{F}_{\lambda_{\text{total}}}$ will output a forgery with probability at least ϵ, thus $\mathcal{F}$ will output a forgery with probability at least $\epsilon \cdot 2^{-\lambda_{\text{total}}}$.

An efficient scheme in the random oracle model. The security loss in the above reduction is exponential in λ_{total}. Recently, Alwen, Dodis and Wichs [2] proposed a signature scheme which can leak a substantial bounded amount λ_{total} of information without suffering an exponential decrease in security. More precisely, [2,19] show that in the random oracle model (a variant of) the Okamoto-Schnorr signature scheme [28,33] is still secure even if a constant fraction λ_{total} of the total secret key is leaked to the adversary. For concreteness we now recall the variant $\mathsf{SIG}_\ell^{\text{OS}} = (\mathsf{Kg}_\ell^{\text{OS}}, \mathsf{Sign}_\ell^{\text{OS}}, \mathsf{Vfy}_\ell^{\text{OS}})$ of the Okamoto-Schnorr signature scheme.

Let $\mathsf{G}(1^k)$ be a group sampling algorithm which outputs a tuple $(p, \mathbb{G})$, where p is a prime of size $\log p = 2k$ and $\mathbb{G}$ is a group of order p in which the discrete logarithm problem is hard.[8] Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function that will be modeled as a random oracle. The scheme is given in Figure 2.

In [2,19] the authors show that $\mathsf{SIG}_\ell^{\text{OS}}$ is secure under the hardness of the ℓ-representation problem (c.f. [2,19] and the references therein for its description and its equivalence to the DL problem). More precisely, they prove the following lemma.

Lemma 2. *For any $\delta > 0$ and $\ell \in \mathbb{N}$, security parameter k, $t = t(k)$, $\epsilon = \epsilon(k)$, $d = d(k)$, $\lambda_{\text{total}} = (1/2 - 1/2\ell - \delta)n$ where $n = 2k\ell$ is the length of the secret key, if the ℓ-representation problem is (t, ϵ)-hard then the signature scheme $\mathsf{SIG}_\ell^{\text{OS}}$*

[8] For technical reasons we assume that elements of $\mathbb{G}$ can be sampled "obliviously", this means, there exists an efficient algorithm $\mathsf{samp}_{\mathbb{G}}$ that outputs 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 $g := \mathsf{samp}_{\mathbb{G}}(\omega)$. See [19] for more details.

```
**Algorithm** $\mathsf{Kg}^{\mathrm{OS}}_{\ell}(1^k)$
    $(\mathbb{G}, p) \xleftarrow{\$} \mathsf{G}(1^k)$
    $(g_1, \ldots, g_\ell) \xleftarrow{\$} \mathbb{G}^\ell$;  $(x_1, \ldots, x_\ell) \xleftarrow{\$} \mathbb{Z}_p^\ell$
    $h \leftarrow \prod_i g_i^{x_i}$
    return $(pk, sk) = ((\mathbb{G}, p, g_1, \ldots, g_\ell, h), (x_1, \ldots, x_\ell))$

**Algorithm** $\mathsf{Sign}^{\mathrm{OS}}_{\ell}(sk, m)$
    $(r_1, \ldots, r_\ell) \xleftarrow{\$} \mathbb{Z}_q^\ell$
    $A \leftarrow \prod_i g_i^{r_i}$
    $c \leftarrow H(A, m)$
    return $\sigma = (A, cx_1 + r_1, \ldots, cx_\ell + r_\ell)$

**Algorithm** $\mathsf{Vfy}^{\mathrm{OS}}_{\ell}(pk, \sigma, m)$
    Parse $\sigma$ as $(A, \alpha_1, \ldots, \alpha_\ell)$
    $c \leftarrow H(A, m)$
    Iff $\prod g_i^{\alpha_i} \overset{?}{=} Ah^c$ return 1; else return 0
```

Fig. 2. $\mathsf{SIG}^{\mathrm{OS}}_{\ell} = (\mathsf{Kg}^{\mathrm{OS}}_{\ell}, \mathsf{Sign}^{\mathrm{OS}}_{\ell}, \mathsf{Vfy}^{\mathrm{OS}}_{\ell})$

from Figure 2 is $(t', d, \epsilon', \lambda_{\text{total}})$ UF-CMTLA secure in the random oracle model, where $t' \approx t$ and $\epsilon' = (q_H \cdot (2 \cdot \epsilon + 1/p + q_H/p^{2\delta\ell}))^{1/2}$, where q_H is the number of random oracle queries made by the adversary.

A SCHEME IN THE STANDARD MODEL. From a universal one-way hash function (UOWHF) H, [19] constructs an efficient *one-time* signature scheme that tolerates leakage of a $(1-\delta)/4$ fraction of the secret key. Using sequential composition this scheme is easily generalized to a stateful d-time signature schemes SIG^K_δ which can leak up to a $(1-\delta)/4d$ fraction of the secret-key.

Lemma 3. *For any $\delta > 0$, security parameter k, $t = t(k)$, $\epsilon = \epsilon(k)$, $d = d(k)$, if H is a (t, ϵ)-secure UOWHF, then SIG^K_δ is $(t', d, \epsilon', \lambda_{\text{total}})$ UF-CMTLA secure, where $\epsilon' = d\epsilon$, $t' \approx t$ and $\lambda_{\text{total}} = n \cdot \frac{1-\delta}{4d}$ where $n = O(dk^2/\delta)$ is the length of the secret key.*

4.2 Construction of Leakage Resilient Signature Schemes

In this section we show how to construct a UF-CMLA secure signature scheme $\mathsf{SIG}^* = (\mathsf{Kg}^*, \mathsf{Sign}^*, \mathsf{Vfy}^*)$ from any UF-CMTLA 3-time signature scheme $\mathsf{SIG} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vfy})$.

We first introduce some notation related to binary trees that will be useful for the description of our signature scheme. For $d \in \mathbb{N}$, we denote with $\{0,1\}^{\leq d} = \bigcup_{i=0}^{d}\{0,1\}^i \cup \varepsilon$ the set of size $2^{d+1}-1$ containing all binary bitstrings of length at most d including the empty string ε. We will think of $\{0,1\}^{\leq d}$ as the labels of a binary tree of depth d. The left and right child of an internal node $w \in \{0,1\}^{\leq d-1}$ are $w0$ and $w1$, respectively. For a node $w \in \{0,1\}^{\leq d} \setminus 1^d$, we denote with $\mathsf{DF}(w)$ the node visited after w in a depth-first traversal.

$$\mathsf{DF}(w) := \begin{cases} w0 & \text{if} \quad |w| < d \qquad\qquad\qquad\quad (w \text{ is an internal node}) \\ \hat{w}1 & \text{if} \quad |w| = d, \text{ where } w = \hat{w}01^t \quad (w \text{ is the root}) \end{cases}$$

We define the mapping $\varphi : \{0,1\}^{\leq d} \to [2^{d-1}-1]$ where $\varphi(w) = i$ if w is the i-th node to be visited in a depth first traversal, i.e. $\varphi(\varepsilon) = 1, \varphi(0) = 2, \varphi(00) = 3, \ldots$.

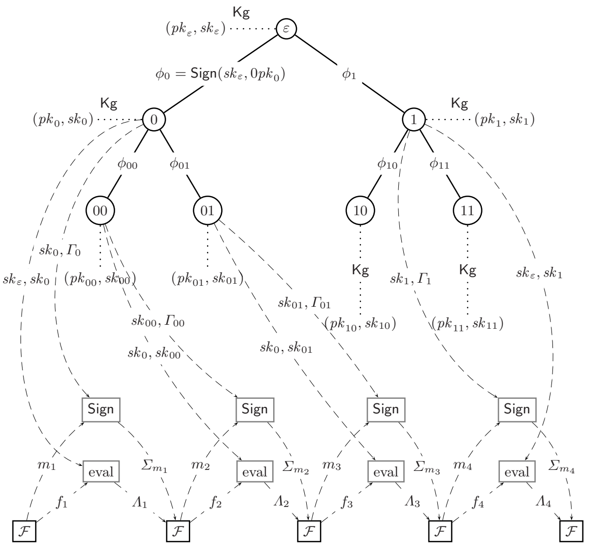

Fig. 3. Illustration of the execution of SIG^* in the UF-CMLA experiment. This figure shows the first 4 rounds of interaction between the adversary $\mathcal{F}$ and Sign. The dotted edges associate a public/secret key to each node. The dashed arrows represent $\mathcal{F}$'s oracle queries. $\mathcal{F}$ queries for a message m_i and a leakage function f_i, and obtains the signature Σ_{m_i}. Additionally, it obtains the leakage function f_i evaluated on the active state S_i^+, which, for instance for $i = 1$, includes the keys sk_ε, sk_0.

We now give the construction of our leakage resilient signature scheme. To simplify the exposition, we will assume that SIG is a stateless signature scheme, but this is not required. We fix some $d \in \mathbb{N}$ such that $q = 2^{d+1} - 2$ is an upper bound on the number of messages that SIG can sign. The signing algorithm Sign^* traverses a tree (depth first), "visiting" the node w and associating to it a key-pair (pk_w, sk_w) generated from the underlying signature scheme SIG. We will use the following notational conventions for a node $w = w_1 w_2 \ldots w_t$.

- $\Gamma_w = [(pk_{w_1}, \phi_{w_1}), (pk_{w_1 w_2}, \phi_{w_1 w_2}), \ldots, (pk_w, \phi_w)]$ is a *signature path* from w to the root, where $\phi_{w'}$ always denotes the signature of $pk_{w'}$ with its parent secret key $sk_{\mathsf{par}(w')}$.
- $S_w = \{sk_{w_1 w_2 \ldots w_i} : w_{i+1} = 0\}$ denotes a subset of the secret keys on the path from the root ε to w. S_w contains $sk_{w'}$, if the path goes to the left child $w'0$ at some node w' on the path. (The reason is, that in this case the right child $w'1$ will be visited after w in a depth first search, and we will then need $sk_{w'}$ to sign the public key $pk_{w'1}$ of that child.)

The secret key of SIG^* will always be of the form (w, S_w, Γ_w), and we will use stacks S and Γ to keep track of the state. We denote an empty stack with $\emptyset$. For a stack A, $\mathsf{push}(A, a)$ denotes putting element a on the stack A, $a \leftarrow \mathsf{pop}(A)$

```
Algorithm $\mathsf{Kg}^*(1^k)$
    $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$
    $S \leftarrow \emptyset$;  $\mathsf{push}(S, sk)$;  $\Gamma \leftarrow \emptyset$
    $SK_0 \leftarrow (w_\varepsilon, S, \Gamma)$;  $PK \leftarrow pk$
    return $(PK, SK_0)$

Algorithm $\mathsf{Vfy}^*(PK, m, \Sigma)$
  parse $\Sigma$ as $(\sigma, \Gamma_{w_1 w_2 \ldots w_t})$
  $pk_\epsilon \leftarrow PK$
  for i = 1 to t do
    if $\mathsf{Vfy}(pk_{w_1 \ldots w_{i-1}}, 0pk_{w_1 \ldots w_i}, \phi_{w_1 \ldots w_i}) = 0$ return 0
  return $\mathsf{Vfy}(pk_{w_1 w_2 \ldots w_t}, 1m, \sigma)$

Algorithm $\mathsf{Sign}^*(SK_i, m)$
  parse $SK_i$ as $(w, S, \Gamma)$                         % then $S = S_w$ and $\Gamma = \Gamma_w$
  if $w = 1^d$ return $\perp$                              % stop if last node reached
  $\hat{w} \leftarrow \mathsf{DF}(w)$                      % compute next node to be visited
  $(sk_{\hat{w}}, pk_{\hat{w}}) \xleftarrow{\$} \mathsf{Kg}(1^n)$     % generate secret key for the current node
  $\sigma \xleftarrow{\$} \mathsf{Sign}(sk_{\hat{w}}, 1m)$            % sign $m$ with secret key of current node
  $sk_{\mathsf{par}(\hat{w})} \leftarrow \mathsf{pop}(S)$             % get secret key of parent (which is on top of $S$)
  $\phi_{\hat{w}} \xleftarrow{\$} \mathsf{Sign}(sk_{\mathsf{par}(\hat{w})}, 0pk_{\hat{w}})$   % sign new $pk$ with $sk$ of its parent
  if $\hat{w}_{|\hat{w}|} = 0$ then $\mathsf{push}(S, sk_{\mathsf{par}(\hat{w})})$   % put $sk_{\mathsf{par}(\hat{w})}$ back if $\hat{w}$ is a left child
  if $|\hat{w}| < d$ then $\mathsf{push}(S, sk_{\hat{w}})$            % put $sk_{\hat{w}}$ on $S$ if it is not a leaf, now $S = S_{\hat{w}}$
  if $|w| = d$                                             % if previous node was a leaf then clean signature chain
    parse $w$ as $w'01^j$
    for $i = 1, \ldots, j+1$ do $\mathsf{trash}(\Gamma)$;
  $\mathsf{push}(\Gamma, (pk_{\hat{w}}, \phi_{\hat{w}}))$            % Now $\Gamma = \Gamma_{\hat{w}}$
  $\Sigma \leftarrow (\sigma, \Gamma)$
  $SK_{i+1} \leftarrow (\hat{w}, S, \Gamma)$                % store key for next signature
  return $(\Sigma, SK_{i+1})$
```

Fig. 4. The leakage resilient signature scheme SIG^*

denotes removing the topmost element from A and assigning it to a, and $\mathsf{trash}(A)$ denotes removing the topmost element from A (without assigning it). To avoid confusion we will always use upper case letters (PK, SK) for keys of SIG^* and lower case letters (pk, sk) for keys used by the underlying signature scheme SIG. To ease exposition, we use the secret key of the node 0, and not the root to sign the first message. The scheme SIG^* is defined in Figure 4.

Theorem 1. *For any security parameter k, $t = t(k)$, $\epsilon = \epsilon(k), q = q(k), \lambda = \lambda(k)$, if SIG is $(t, 3, \epsilon, \lambda_{\text{total}})$ UF-CMTLA secure, then SIG^* is $(t', q-1, q\epsilon, \lambda)$ UF-CMLA secure where $t' \approx t$ and $\lambda = \lambda_{\text{total}}/3$.*

Proof. We will show how to construct an adversary $\mathcal{F}$ which breaks the UF-CMTLA security of SIG (with $\lambda_{\text{total}} = 3 \cdot \lambda$ bits of total leakage) using as a subroutine the adversary $\mathcal{F}_\lambda$ who breaks the UF-CMLA security of SIG^* (with λ bits of leakage in each of the q observations) with advantage at least

$$\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cmtla}}(\mathcal{F}, k, \lambda_{\text{total}}) \geq \frac{1}{q} \cdot \mathbf{Adv}_{\mathsf{SIG}^*}^{\text{uf-cmla}}(\mathcal{F}_\lambda, k, \lambda) . \tag{1}$$

The adversary $\mathcal{F}(pk)$ (attacking the UF-CMTLA security of SIG) simulates $\mathcal{F}_\lambda(PK)$ attacking the UF-CMLA security of SIG^*, embedding its challenge public-key pk into one of the nodes of SIG^*. That is, $\mathcal{F}(pk)$ simulates the following experiment (as defined in Section 3.2, cf. also Figure 3 for a graphical illustration.)

```
Experiment $\mathbf{Exp}^{\text{uf-cmla}}_{\mathsf{SIG}^*}(\mathcal{F}_\lambda, k, \lambda)$
    $(PK, SK_0) \overset{\$}{\leftarrow} \mathsf{Kg}^*(1^k)$ ; $i \leftarrow 1$
    $(m, \Sigma) \overset{\$}{\leftarrow} \mathcal{F}_\lambda^{\mathcal{O}_{SK_{i-1}}(\cdot,\cdot)}(PK)$
    If $\mathsf{Vfy}^*(PK, m, \Sigma) = 1$
        and $m \notin \{m_1, \ldots m_i\}$
        then return 1 else return 0.

Oracle $\mathcal{O}_{SK_{i-1}}(m_i, f_i)$
    Sample fresh randomness $r_i$
    $(\Sigma_i, SK_i) \overset{SK^+_{i-1}}{\hookleftarrow} \mathsf{Sign}^*(SK_{i-1}, m_i, r_i)$
    $\Lambda_i \leftarrow f_i(SK^+_{i-1}, r_i)$
    if $|\Lambda_i| \neq \lambda$ then $\Lambda_i \leftarrow 0^\lambda$
    Return $(\Sigma_i, \Lambda_i)$ and set $i \leftarrow i + 1$
```

Simulation of PK. On input pk, $\mathcal{F}$ samples a node $\tilde{w}$ at random from the first q nodes (i.e., $\tilde{i} \overset{\$}{\leftarrow} \{1, \ldots, q\}$ and $\tilde{w} \leftarrow \varphi^{-1}(\tilde{i})$). The key $(pk_{\tilde{w}}, sk_{\tilde{w}})$ used by Sign will be the challenge key (pk, sk). Note that $sk = sk_{\tilde{w}}$ is unknown to $\mathcal{F}$. Next, $\mathcal{F}$ generates the other keys $(pk_w, sk_w), w \in \{0,1\}^{\leq d} \setminus \tilde{w}$ by calling $\mathsf{Kg}(1^k)$ using fresh randomness for each call. (Of course, these keys will only be computed when needed during the simulation of the signing oracle.) $\mathcal{F}$ defines $PK = pk_\varepsilon$ and runs $\mathcal{F}_\lambda$ on PK.

Simulation of the signing oracle. Let (m_i, f_i) be the i-th query to oracle $\mathcal{O}_{SK_{i-1}}(m_i, f_i)$ and let SK^+_{i-1} be the active state information in an execution of the real signing algorithm (i.e., $(\Sigma_i, SK_i) \overset{SK^+_{i-1}}{\hookleftarrow} \mathsf{Sign}^*(SK_{i-1}, m_i, r_i)$). Depending if $sk_{\tilde{w}} \in SK^+_{i-1}$ or not, adversary $\mathcal{F}$ distinguishes the two cases.

Case 1: $sk_{\tilde{w}} \notin SK^+_{i-1}$ ($\mathsf{Sign}(SK_{i-1}, m_i, r_i)$ does not access $sk_{\tilde{w}}$.) In this case the adversary $\mathcal{F}$ computes $\sigma_i \overset{\$}{\leftarrow} \mathsf{Sign}(SK_{i-1}, m_i, r_i)$ and $\Lambda_i = f_i(SK^+_{i-1}, r_i)$ itself and outputs (σ_i, Λ_i).

Case 2: $sk_{\tilde{w}} \in SK^+_{i-1}$ ($\mathsf{Sign}(SK_{i-1}, m_i, r_i)$ does access $sk_{\tilde{w}} \in SK^+_{i-1}$.) In this case $\mathcal{F}$ can compute (σ_i, Λ_i) without knowing $sk_{\tilde{w}} = sk$ as it has access to the signing oracle $\mathcal{O}_{sk_{\tilde{w}}}$ and the leakage oracle $\mathcal{O}_{\text{leak}}$ as defined in the CMTLA attack game. As $sk_{\tilde{w}} \in SK^+_{i-1}$ for at most three different i, and on for each i the range of f_i is λ bits, the total leakage will be $\lambda_{\text{total}} = 3 \cdot \lambda$ bits, which is what we assume $\mathcal{F}$ can get from $\mathcal{O}_{\text{leak}}$.

The simulation of the UF-CMLA experiment by $\mathcal{F}$ is perfect (i.e. has the right distribution). As $\mathcal{F}$ perfectly simulates the UF-CMLA experiment, by assumption, $\mathcal{F}_\lambda$ does output a forgery with probability $\mathbf{Adv}^{\text{uf-cmla}}_{\mathsf{SIG}}(\mathcal{F}_\lambda, k, \lambda)$. We now show that from $\mathcal{F}$'s forgery one can extract a forgery for at least one of the keys (pk_w, sk_w) of the underlying signature scheme SIG.

Claim. If $\mathcal{F}_\lambda$ outputs a forgery (σ, Σ) in the UF-CMLA experiment, then one can extract a forgery for SIG with respect to at least one of the public-keys $(pk_w, sk_w), w \in \{\varphi_d^{-1}(1), \ldots, \varphi_d^{-1}(q)\}$.

Proof. Let $W = \{\varphi_d^{-1}(0), \ldots, \varphi_d^{-1}(q)\}$ be the set of nodes that have been visited during the query phase of the UF-CMLA experiment. Further, let $U := \{\Gamma_w\}_{w \in W}$ be the set of all signature chains that have been generated during the experiment. We distinguish two cases.

Case 1: $\Gamma \in U$. Then $\Gamma = \Gamma_w$ for one $w \in W$. If $\Sigma = (\sigma, \Gamma)$ is a valid forgery, then $\sigma \in \mathsf{Sign}(sk_w, 1m)$, where $m \neq m_{\varphi_d^{-1}(w)}$. Thus, $(1m, \sigma)$ is a valid forgery of SIG for public key pk_w.

Case 2: $\Gamma \notin U$. Then there must exist a node $w \in W$ such that $\phi \in \Gamma$ with $\phi \in \mathsf{Sign}(sk_w, 0pk_{w^*})$, where $pk_{w^*} \neq pk_{w0}$ and $pk_{w^*} \neq pk_{w1}$.[9] It follows that ϕ is a valid signature for key pk_w and message $0pk_{w^*}$ that has not been queried before.

The claim follows. $\triangle$

With this claim and the fact that the simulation is perfect, it follows that we can extract a forgery for SIG with respect to the challenge public-key pk with probability $\mathbf{Adv}^{\text{uf-cmla}}_{\mathsf{SIG}^*}(\mathcal{F}_\lambda, k, \lambda)/q$ (namely when the w from the claim is $\tilde{w}$). This proves (1) and completes the proof. $\square$

4.3 Efficiency and Trade-offs

We analyze the performance of our basic leakage resilient signature scheme and provide some efficiency trade-offs. For $d \in \mathbb{N}$ let $D = 2^{d+1} - 2$ be the upper bound on the number of messages that can be signed.

For simplicity, we assume that for SIG key generation, signing and verification all take approximately the same time, and further that public keys, secret keys and signatures are all of the same length. Let us now analyze the efficiency of SIG^*. Public key size and key generation are as in the underlying scheme.

In the signing process, Sign^* has to run at most two instances of Sign (i.e., to sign the message and to certify the next public key) and one run of the underlying key generation algorithm Kg. This adds up to an overhead of 3 compared to SIG. In our scheme, a signature consists of the signature of the actual message together with a signature chain from the current signing node to the root. Thus, the size of a signature increases in the worst case (if we sign with a leaf node) by a factor of $\approx 2d$. For the verification of a signature, in Vfy^* we have to first verify the signature chain, and only if all those verifications pass, we check the signature on the actual message. This results in an overhead of d compared to the the underlying verification algorithm Vfy. Finally, in contrast to SIG our scheme requires storage of $\approx d$ secret keys, $\approx d$ public keys and $\approx d$ signatures, whereas in a standard signature scheme one only has to store a single secret key. Note however that only the storage for the secret keys needs to be kept secret.

In the special case, when we instantiate SIG^* with $\mathsf{SIG}^{\mathrm{OS}}$ and set $\delta = 1/2$ (thus, $\ell = 3$), then SIG^* is quite efficient[10]: signing requires only 9 exponentiations and 2 evaluations of a hash function. Verification is slightly less efficient and needs in the worst case $4d$ exponentiations and d evaluations of the underlying hash function. Finally, in the worst case a signature contains $18d$ group elements. Notice that our construction instantiated with $\mathsf{SIG}^{\mathrm{OS}}$ allows us to leak a 1/36th fraction of the secret key in each observation. It is easy to increase this to a 1/24th fraction by only using the leafs of SIG^* to sign actual messages.

[9] Wlog assume that $w0$ and $w1$ are both in W.

[10] Only counting exponentiations and hash function evaluations.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
3. Anderson, R.: Two remarks on public-key cryptology. Manuscript. Relevant material presented by the author in an invited lecture at the 4th ACM Conference on Computer and Communications Security, CCS 1997, Zurich, Switzerland, April 1–4, September 2000 (1997)
4. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
5. Crescenzo, G.D., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)
6. Dodis, Y., Goldwasser, S., Kalai, Y., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978. Springer, Heidelberg (2010)
7. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: 41st ACM STOC. ACM Press, New York (2009)
8. Dziembowski, S.: On forward-secure storage (extended abstract). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 251–270. Springer, Heidelberg (2006)
9. Dziembowski, S., Pietrzak, K.: Intrusion-resilient secret sharing. In: 48th FOCS, pp. 227–237. IEEE Computer Society Press, Los Alamitos (2007)
10. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS 2008. ACM Press, New York (2008)
11. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.: Leakage-resilient signatures. Cryptology ePrint Archive, Report 2009/282 (2009), http://eprint.iacr.org/
12. Faust, S., Reyzin, L., Tromer, E.: Protecting circuits from computationally-bounded leakage. Cryptology ePrint Archive, Report 2009/379 (2009), http://eprint.iacr.org/
13. Fouque, P.-A., Martinet, G., Poupard, G.: Attacking unbalanced RSA-CRT using SPA. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 254–268. Springer, Heidelberg (2003)
14. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
15. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
16. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
17. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX Security Symposium, pp. 45–60 (2008)
18. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)

19. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
20. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
21. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
22. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (October 1979)
23. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
24. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
25. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
26. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: 21st ACM STOC, pp. 33–43. ACM Press, New York (1989)
27. European Network of Excellence (ECRYPT). The side channel cryptanalysis lounge, http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html (retrieved on 29.03.2008)
28. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
29. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
30. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, I., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
31. Quisquater, J.-J., Koene, F.: Side channel attacks: State of the art. [27](October 2002)
32. Schindler, W.: A timing attack against RSA with the chinese remainder theorem. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 109–124. Springer, Heidelberg (2000)
33. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
34. Standaert, F.-X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)