Leakage-Resilient Signatures

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

[1] K.U. Leuven ESAT-COSIC, 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 of the scheme's hardware implementation. 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 possible* 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*. Based on recent works by Alwen, Dodis, Wichs and by Katz we give several efficient instantiations of 3-time signature schemes with the required security properties, hence yielding the first constructions of provably secure leakage-resilient signature schemes.

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 box stays secret. In particular, the standard security notion of digital signatures is existential unforgeability under chosen message attacks [18] (UF-CMA), where one requires than 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 [26], electromagnetic radiation [38, 17], power consumption [27], fault detection [7, 6], and many more (see, e.g., [39, 34]).

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

⋆ Supported in part by Microsoft Research through its PhD Scholarship Programme. 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.

specific new attack. Their inherent limitation, however, is that they must be tailored specifically for the class of attacks they intend to defeat. Not very surprisingly, these countermeasures often were later found to be insufficient. This 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[4] and for a broad and well-defined attack scenario. (E.g., existential unforgeability for signature schemes or IND-CCA2 security for encryption.)

A FORMAL SECURITY DEFINITION. Recently, a notion of *leakage-resilience* was proposed that adapts the methodology of modern cryptography to the scenario of side-channel attacks [15]. 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 security experiment. The side-channel information given to the adversary only has to satisfy the following two "leakage 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 $f(s) = s$, no security whatsoever can be achieved. We chose to bound the output length of the leakage functions as it is a natural resource bound and allows to model many side-channel attacks (e.g. timing or hamming-weight attacks, which exploit only a polylogarithmic amount of information on each invocation, which is far below the constant fraction for which we can still prove security.) Let us mention that technically, the restriction we must make on the leakage functions is much weaker than a bound on the leakage function,[5] but we will stick to *bounded leakage* (LR1) which is more intuitive and simpler to work with.

Unfortunately bounded leakage alone is not *sufficient*,[6] thus one has to restrict the leakage functions further. The additional restriction should be such that it has a natural interpretation on the implementation level which seem realistic to satisfy, and of course should be strong enough as to admit provably secure leakage-resilient primitives. Following [15], we use LR2 ("only computation leaks information"), originally put forward as one of the "axioms" of "physically observable cryptography" by Micali and Reyzin [31]. On the implementation level, it states that if a prim-

[4] In complexity based cryptography one always 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.

[5] 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.

[6] To see this, let s_i denote the state of the primitive (say a signature scheme) after the ith invocation. We could have leakage functions $f_1, f_2, \ldots, f_t$ where each $f_i(s_i)$ first computes the future state s_t and output a few bits of it. If we choose t large enough, we can learn the entire state s_t even if each f_i leaks only one bit. Note that the above argument only works if we assume that one cannot sample randomness, as otherwise the state update could be probabilistic. We don't have such an impossibility result for the setting where true randomness is available, but it seems extremly hard to achieve anything assuming only bounded leakage.

itive with secret internal state s is invoked, then on this particular invocation, only the parts of the memory leak, which are accessed during this invocation. We refer the reader to [31, 15, 37] for further discussion and motivation on this axiom.

1.2 Leakage-Resilient Signatures

The only primitives known to date that are provably leakage-resilient in the standard model are symmetric stream-ciphers [15, 37]. Very recently, a leakage-resilient public-key encryption scheme [25] was proposed which has a security proof in the generic group model. In this paper we construct the first leakage-resilient public-key primitive in the plain model, a signature-scheme. The techniques we use are completely different from the construction in [25], informally, in [25] leakage-resilience is achieved by *sharing* the secret key, whereas we achieve leakage-resilience by *evolving* the key. Digital signatures are one of the most fundamental cryptographic primitives that are often implemented on devices that are potentially exposed to side-channel attacks (such as smart cards). Starting with the seminal work by Kocher [26], there have been a great number of theoretical and practical side-channel attacks on signature schemes (e.g., [26, 27, 41, 16]).

Security Definition. The standard notion for secure signatures schemes is that of unforgeability under adaptive chosen-message attacks [18]. 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. We call this notion λ-leakage resilience or unforgeability under chosen-message and leakage attacks. 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 messag/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. Based on the ideas of Lamport [28] and Merkle [30], 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 strongly resembles the construction of a forward-secure signature scheme [3] from [5], 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 [35, 42] 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 [24] 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 [24], Katz also constructs 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.[7] This construction is prefect for our purpose and gives 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) on each invocation. As the construction only assumes universal one-way hash functions (UOWHF), we get a security proof in the standard model under the minimal [33] assumption that one-way functions exist.

1.3 Deterministic Leakage-Resilient Signatures

In the construction of SIG^* we silently assumed that the key-generation and signing algorithms of SIG can sample uniform random bits. However, for this one requires some special hardware like noise generating gates. Normally, one can easily avoid the necessity for such special hardware by using pseudorandomness instead of truly random bits by generating the randomness using a stream

[7] Katz proposes 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 $\Omega(n/12)$). The bound of [24] is worse as it aims for a stateless scheme, whereas we do not care about state, as our construction is stateful anyway.

cipher. In our leakage setting, however, it is not obvious if this approach does not compromise security. In particular, using a standard stream-cipher could make our signature scheme insecure against UF-CMLA attacks.[8]

The "obvious" solution here seems to be using a *leakage-resilient* stream cipher [15, 37], but it's not clear how to prove this. Roughly, the problem is that the output of a leakage-resilient stream cipher is indistinguishable from some distribution with high min-entropy (when given the leakage). Unfortunately a signature scheme which is UF-CMTLA secure with λ_{total} bits of leakage does not imply that the scheme will still be secure if the radomness R used in the scheme is sampled from some distribution with min-entropy $|R| - \lambda_{\text{total}}$.[9] Thus, to achieve UF-CMLA security for SIG^* where we generate all the randomness using a leakage-resilient stream-cipher (we'll refer to this construction as SIG^{**}), UF-CMTLA of the underlying 3-times signature scheme SIG is not enough.

Random Oracle Model. In the random-oracle model, this problem can be trivially solved by applying a random oracle $\mathcal{H}$ to the outputs $X_1, X_2, \ldots$ of the leakage-resilient stream cipher. That is, one uses $Y_i \leftarrow \mathcal{H}(X_i)$ as randomness for the ith signature query. The reason this works follows from the simple fact that if X_i is unpredicteable, then $\mathcal{H}(X_i)$ is uniformly random (and not only pseudorandom) given the view of the adversary.

2-Source Extractors. An approach which looks promising (but which we did not work out) to achieve a deterministic instantiation in the standard model is to use a 2-source extractor ext (cf.[40] and references therein) as follows: for each invocation of the signature scheme one invokes the leakage-resilien stream cipher twice to get X and X', and then extract randomness $Y \leftarrow \mathsf{ext}(X, X')$. By definition, $\mathsf{ext}(A, B)$ is uniformly random whenever A and B have sufficiently high min-entropy and are independent.

Generic Result loosing an Exponential Factor. In Appendix A we prove that SIG^{**} is UF-CMLA secure for any underlying 3-times signature scheme SIG, but the loss in security is exponential in the leakage λ. Thus, if we only assume security of SIG against poly-size adversaries, we can only tolerate leakage which is logarithmic in the length of the accessed state. To leak a constant fraction, we'd have to make exponential hardness assumptions.

Let us remark that the known constructions of leakage-resilient stream ciphers [15, 37] need exponential hardness assumptions to tolerate a constant fraction of leakage, so why not make such assumptions on SIG also. The reason is that stream-ciphers are constructed from symmetric primitiev (e.g., [37] can be directly instantited with any block-cipher like AES), and it is generally believed that we can construct symmetric primites which are efficient and exponentially hard to break (e.g., no attacks faster than brute force-key search against AES are known). On the other hand, *efficient* constructions of signature schemes rely on very special assumptios, e.g., Okamoto-Schnorr relies on the hardness of the discrete logarithm problem, which are either known not to have exponential hardness, or such an assumption seems pretty strong.

1.4 Related Work

Most algorithmic countermeasures for side-channel attacks only consider some particular side-channels. For example Ishai et al. [22, 21] show how to securely implement any function if the

[8] E.g. using a block-cipher in CBC mode is trivially insecure, as here the block-cipher key is fixed and thus can be completely leaked.

[9] Only the other direction is true: if X is uniform, then for any function f with range $\{0,1\}^{\lambda_{\text{total}}}$, X has (expected) min-entropy $|X| - \lambda_{\text{total}}$ given $f(X)$.

attacker can probe a bounded number of wires. This work is remarkable as it was the first to show how to implement *any* algorithm in a way that is *provably secure* against an interesting side-channel (i.e., probing attacks).

Micali and Reyzin [31] in their work on "physically observable cryptography" proposed an influential theoretical framework to model side-channel attacks. In particular, they state and motivate the "only computation leaks information" axiom used in leakage-resilient cryptography [15, 37, 25]. Standaert et al. [43] consider a restricted version of the [31] model which still captures reasonable adversaries and leakage functions that have been successfully used in practice to break systems. In this model Petit et al. [36] analyze a block-cipher based construction for a PRNG (cf. [37] on how this notion compares to the leakage-resilient notion).

The other restriction (besides "only computation leaks information") that is used in the leakage-resilient framework is a bound on the output length of the leakage functions, this is inspired by the bounded-retrieval model [9, 12, 11, 8, 14, 23] which in turn was inspired by the bounded-storage model [29, 13, 44, 19].

Several recent works [1, 32, 24, 10] achieve constructions which are leakage-resilient against general leakage functions without relying on the "only computation leaks information" axiom. All this constructions are stateless, and thus cannot tolerate any kind of continuous leakage. In particular, Akavia et. al [1] and Naor and Segev [32] construct public-key encryption schemes that remain secure even if a function $f(sk)$ of the secret key is leaked. The function $f(\cdot)$ can be arbitrary, but must have bounded range of λ bits (where $\lambda \ll |sk|$). We already discussed the work of Katz [24] who constructs digital signatures in this setting. This setting can be seen as a restricted version of intrusion-resilience as discussed in the work of Alwen et al. [23], who construct intrusion-resilient identification protocols. Dodis et al. [10] consider the case where the range of $f(\cdot)$ is not necessarily bounded, but instead one only requires that it is (exponentilly) hard to recover sk from $f(sk)$.

2 Preliminaries

2.1 Notation

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 $[n]$ shorthand for $\{1, \ldots, n\}$. If S is a set then $s \xleftarrow{\$} S$ denotes the operation of picking an element s of S uniformly at random. With PPT we denote probabilistic polynomial time.

2.2 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, $y \xleftarrow{\$} \mathcal{A}(x)$ denotes runnig 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 exectution 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.[10] 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'.

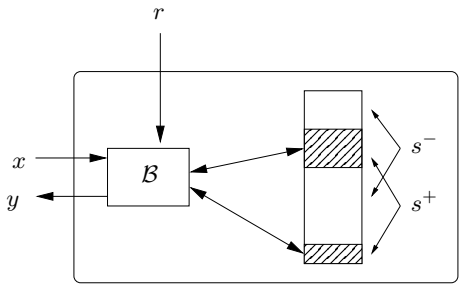

Fig. 1. Illustration of the execution of a stateful algorithm $(y, s') \overset{s^+}{\hookleftarrow} \mathcal{B}(x, s, r)$. The secret state s splits into the active state s^+ (that is accessed during the execurition of $\mathcal{B}$) and the passive state s^-.

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

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

[10] 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.

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 $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{SIG}}(\mathcal{F}, k)$ we denote the probability that the above experiment returns 1. Forger $\mathcal{F}$ (t, q, ϵ)-breaks the UF-CMA security of SIG if $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{SIG}}(\mathcal{F}, k) \geq \epsilon$, 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 (given by a circuit) 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 $\mathbf{Exp}^{\text{uf-cmla}}_{\mathsf{SIG}}(\mathcal{F}, k, \lambda)$
$(PK, SK_0) \xleftarrow{\$} \mathsf{Kg}(1^k)$; $i \leftarrow 1$
$(m^*, \sigma^*) \xleftarrow{\$} \mathcal{F}^{\mathcal{O}_{SK_{i-1}}}(PK)$
If $\mathsf{Vfy}(PK, m^*, \sigma^*) = 1$ and $m^* \notin \{m_1, \ldots m_i\}$
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 (σ_i, Λ_i) and set $i \leftarrow i + 1$

With $\mathbf{Adv}^{\text{uf-cmla}}_{\mathsf{SIG}}(\mathcal{F}, k, \lambda)$ 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 $\mathbf{Adv}^{\text{uf-cmla}}_{\mathsf{SIG}}(\mathcal{F}, k, \lambda) \geq \epsilon(k)$. 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 [24], 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 $\mathbf{Exp}^{\text{uf-cmtla}}_{\mathsf{SIG}}(\mathcal{F}, k, \lambda_{\text{total}})$
$(pk, sk_0) \xleftarrow{r_0} \mathsf{Kg}(1^k)$; $i \leftarrow 1$; $\lambda_{\text{cnt}} \leftarrow 0$; $\mathsf{state} \leftarrow r_0$
$(m^*, \sigma^*) \xleftarrow{\$} \mathcal{F}^{\mathcal{O}_{sk_{i-1}}, \mathcal{O}_{\text{leak}}}(pk)$
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
$\mathsf{state} \leftarrow \mathsf{state} \cup r_i$
$(\sigma_i, sk_i) \leftarrow \mathsf{Sign}(sk_{i-1}, m_i, r_i)$
Return σ_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 Λ

With $\mathbf{Adv}^{\text{uf-cmtla}}_{\mathsf{SIG}}(\mathcal{F}, k, \lambda_{\text{total}})$ 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 $\mathbf{Adv}^{\text{uf-cmtla}}_{\mathsf{SIG}}(\mathcal{F}, k, \lambda_{\text{total}}) \geq \epsilon(k)$. 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. Suppose 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 $\mathcal{F}^{\mathcal{O}_{sk_{i-1}}, \mathcal{O}_{\text{leak}}}_{\lambda_{\text{total}}}(pk)$, 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 $2^{-\lambda_{\text{total}}}$. Conditionned 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, 24] show that in the random oracle model (a variant of) the Okamoto-Schnorr signature scheme [35, 42] 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 $\mathsf{SIG}_\ell^{\mathrm{OS}} = (\mathsf{Kg}_\ell^{\mathrm{OS}}, \mathsf{Sign}_\ell^{\mathrm{OS}}, \mathsf{Vfy}_\ell^{\mathrm{OS}})$ 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.[11] 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.

Algorithm $\mathsf{Kg}_\ell^{\mathrm{OS}}(1^k)$	**Algorithm** $\mathsf{Sign}_\ell^{\mathrm{OS}}(sk, m)$
$(\mathbb{G}, p) \xleftarrow{\$} \mathsf{G}(1^k)$ $(g_1, \ldots, g_\ell) \xleftarrow{\$} \mathbb{G}^\ell$; $(x_1, \ldots, x_\ell) \xleftarrow{\$} \mathbb{Z}_p^\ell$ $h \leftarrow \prod_i g_i^{x_i}$ return $(pk, sk) = ((\mathbb{G}, p, g_1, \ldots, g_\ell, h), (x_1, \ldots, x_\ell))$	$(r_1, \ldots, r_\ell) \xleftarrow{\$} \mathbb{Z}_q^\ell$ $A \leftarrow \prod_i g_i^{r_i}$ $c \leftarrow H(A, m)$ return $\sigma = (A, cx_1 + r_1, \ldots, cx_\ell + r_\ell)$
Algorithm $\mathsf{Vfy}_\ell^{\mathrm{OS}}(pk, \sigma, m)$ Parse σ as $(A, \alpha_1, \ldots, \alpha_\ell)$ $c \leftarrow H(A, m)$ Iff $\prod g_i^{\alpha_i} \stackrel{?}{=} Ah^c$ return 1; else return 0	

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

We now state the hardness assumptions. We say that the DL problem is (t, ϵ)-hard if for every adversary A running in time at most t the probability $\mathsf{Pr}[x \xleftarrow{\$} \mathsf{A}(\mathbb{G}, p, g^x, \omega) \mid (\mathbb{G}, p) \xleftarrow{\$} \mathsf{G}(1^k), x \xleftarrow{\$} \mathbb{Z}_p]$ is negligible in k, where ω are the random coins used to generate A's input.

In order to give a concrete security result we also need to introduce the ℓ-representation problem. We say that the ℓ-representation problem is (t, ϵ)-hard for $\mathbb{G}$ if for every adversary A running in time at most t the probability

$$\mathsf{Pr}\left[\prod g_i^{x_i} = \prod g_i^{x'_i} \ \wedge\ (x_1, \ldots, x_\ell) \neq (x'_1, \ldots x'_\ell) \;\middle|\; \begin{array}{l}(\mathbb{G}, p) \xleftarrow{\$} \mathsf{G}(1^k);\ \ g_1, \ldots, g_\ell \xleftarrow{\$} \mathbb{G};\\ (x_1, \ldots, x_\ell, x'_1, \ldots x'_\ell) \xleftarrow{\$} \mathsf{A}(\mathbb{G}, p, g_1, \ldots, g_\ell, \omega)\end{array}\right]$$

is negligible in k, where ω is the randomness used to sample A's input. Hardness of the discrete logarithm problem for $\mathbb{G}$ implies hardness of the ℓ-representation problem (for any polynomial ℓ).

In [2, 24] the following lemma is proved.

Lemma 2. *For any $\delta > 0$ and $\ell \in \mathbb{N}$, security parameter k, $t = t(k)$, $\epsilon = \epsilon(k)$, $d = d(k)$, $\lambda_{\mathrm{total}} = (1/2 - 1/2\ell - \delta)n$ where $n = 2k\ell$ is the length of the secret key, if the ℓ-representation problem is (t, ϵ)-hard then the signature scheme $\mathsf{SIG}_\ell^{\mathrm{OS}}$ from Figure 2 is $(t', d, \epsilon', \lambda_{\mathrm{total}})$ 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.*

We remark that [24] also discusses efficient schemes based on the RSA or factoring assumptions in the random oracel model.

A SCHEME IN THE STANDARD MODEL. From a universal one-way hash function (UOWHF) H, Katz constructs an efficient *one-time* signature scheme that tolerates leakage of a $(1 - \delta)/4$ fraction of

[11] 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 [24] for more details.

the secret key. Using sequential composition this scheme is easily generalized to a stateful d-time signature schemes SIG_δ^K 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_δ^K 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.

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

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. $\varphi(\varepsilon) = 1, \varphi(0) = 2, \varphi(00) = 3, \ldots$.

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, $a \leftarrow \mathsf{pop}(A)$ 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 3.

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 $\lambda = \lambda_{\text{total}}/3$.*

Algorithm $\mathsf{Kg}^*(1^k)$	**Algorithm** $\mathsf{Vfy}^*(PK, m, \Sigma)$
$(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$	parse Σ as $(\sigma, \Gamma_{w_1 w_2 \ldots w_t})$
$S \leftarrow \emptyset$; $\mathsf{push}(S, sk)$; $\Gamma \leftarrow \emptyset$	$pk_\epsilon \leftarrow PK$
$SK_0 \leftarrow (w_\varepsilon, S, \Gamma)$; $PK \leftarrow pk$	for i = 1 to t do
return (PK, SK_0)	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, Γ) % 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 (Σ, SK_{i+1})

Fig. 3. The leakage resilient signature scheme SIG^*.

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

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

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 challange 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 5 for a graphical illustration.)

Experiment $\mathbf{Exp}^{\text{uf-cmla}}_{\mathsf{SIG}^*}(\mathcal{F}_\lambda, k, \lambda)$	**Oracle** $\mathcal{O}_{SK_{i-1}}(m_i, f_i)$
$(PK, SK_0) \xleftarrow{\$} \mathsf{Kg}^*(1^k)$; $i \leftarrow 1$	Sample fresh randomness r_i
$(m, \Sigma) \xleftarrow{\$} \mathcal{F}_\lambda^{\mathcal{O}_{SK_{i-1}}(\cdot,\cdot)}(PK)$	$(\Sigma_i, SK_i) \overset{SK^+_{i-1}}{\hookleftarrow} \mathsf{Sign}^*(SK_{i-1}, m_i, r_i)$
If $\mathsf{Vfy}^*(PK, m, \Sigma) = 1$ and $m \notin \{m_1, \ldots m_i\}$	$\Lambda_i \leftarrow f_i(SK^+_{i-1}, r_i)$
then return 1 else return 0.	if $\lvert\Lambda_i\rvert \neq \lambda$ then $\Lambda_i \leftarrow 0^\lambda$
	Return (Σ_i, Λ_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., $\tilde{i} \xleftarrow{\$} \{1, \ldots, q\}$ 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 $\mathbf{Adv}_{\mathsf{SIG}}^{\text{uf-cmla}}(\mathcal{F}_\lambda, k, \lambda)$. 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 $pk_{w^*} \neq pk_{w1}$.[12] 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. △

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 $\mathbf{Adv}_{\mathsf{SIG}^*}^{\text{uf-cmla}}(\mathcal{F}_\lambda, k, \lambda)/q$ (namely when the w from the claim is $\tilde{w}$). This proves (1) and completes the proof. □

4.3 Efficiency and Tradeoffs

We analyze the performance of our basic leakage resilient signature scheme and provide some efficiency tradeoffs. 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

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

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[13]: 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.

References

1. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, 2009.
2. Joel Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage resilient public-key cryptography in the bounded retrieval model. In *Advances in Cryptology — CRYPTO 2009*, 2009. to appear.
3. Ross Anderson. 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, 1997, September 2000.
4. Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *RANDOM-APPROX*, pages 200–215, 2003.
5. Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer-Verlag, Berlin, Germany, August 1999.
6. Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer-Verlag, Berlin, Germany, August 1997.
7. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer-Verlag, Berlin, Germany, May 1997.
8. David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 479–498. Springer-Verlag, Berlin, Germany, February 2007.
9. Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 225–244. Springer-Verlag, Berlin, Germany, March 2006.
10. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *41st ACM STOC*. ACM Press, 2009.
11. Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 207–224. Springer-Verlag, Berlin, Germany, March 2006.

[13] only counting exponentiations and hash function evaluations.

12. Stefan Dziembowski. On forward-secure storage (extended abstract). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 251–270. Springer-Verlag, Berlin, Germany, August 2006.
13. Stefan Dziembowski and Ueli M. Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology*, 17(1):5–26, January 2004.
14. Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *48th FOCS*, pages 227–237. IEEE Computer Society Press, October 2007.
15. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS 2008*. ACM Press, 2008.
16. Pierre-Alain Fouque, Gwenaëlle Martinet, and Guillaume Poupard. Attacking unbalanced RSA-CRT using SPA. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 254–268. Springer-Verlag, Berlin, Germany, September 2003.
17. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *CHES*, pages 251–261, 2001.
18. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
19. Danny Harnik and Moni Naor. On everlasting security in the hybrid bounded storage model. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 192–203. Springer-Verlag, Berlin, Germany, July 2006.
20. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
21. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 308–327. Springer-Verlag, Berlin, Germany, May / June 2006.
22. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer-Verlag, Berlin, Germany, August 2003.
23. Yevgeniy Dodis Joel Alwen and Daniel Wichs. Leakage resilient public-key cryptography in the bounded retrieval model. In *CRYPTO*, 2009.
24. Jonathan Katz. Signature schemes with bounded leakage resilience. Cryptology ePrint Archive, Report 2009/220, 2009. http://eprint.iacr.org/.
25. Eike Kiltz and Krzysztof Pietrzak. How to secure elgamal against side-channel attacks. Cryptology ePrint Archive, Report 2009/???, 2009. http://eprint.iacr.org.
26. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, Berlin, Germany, August 1996.
27. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, Berlin, Germany, August 1999.
28. Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
29. Ueli M. Maurer. A provably-secure strongly-randomized cipher. In Ivan Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 361–373. Springer-Verlag, Berlin, Germany, May 1990.
30. Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer-Verlag, Berlin, Germany, August 1990.
31. Silvio Micali and Leonid Reyzin. Physically observable cryptography. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer-Verlag, Berlin, Germany, February 2004.
32. Gil Segev Moni Naor. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, 2009.
33. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.
34. 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.
35. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer-Verlag, Berlin, Germany, August 1993.
36. Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *ASIACCS*, pages 56–65, 2008.
37. Krzysztof Pietrzak. A leakage-resilient mode of operation. In *Eurocrypt*, pages 462–482, 2009.
38. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.

39. Jean-Jaques Quisquater and Franois Koene. Side channel attacks: State of the art, October 2002. [34].
40. Anup Rao. An exposition of bourgain's 2-source extractor. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(034), 2007.
41. Werner Schindler. A timing attack against RSA with the chinese remainder theorem. In Çetin Kaya Koç and Christof Paar, editors, *CHES 2000*, volume 1965 of *LNCS*, pages 109–124. Springer-Verlag, Berlin, Germany, August 2000.
42. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer-Verlag, Berlin, Germany, August 1990.
43. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, pages 443–461, 2009.
44. Salil P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 61–77. Springer-Verlag, Berlin, Germany, August 2003.

A Deterministic Instantiation

SIG^* assumes that its implementation has some means of sampling uniformly random bits. As discussed in Section 1.3, in this section we prove that one can use a leakage-resilient stream cipher to generate this randomness. This makes our construction deterministic and, in particular avoids the necessity for special randomness generating hardware.

A.1 Leakage-Resilient Stream-Ciphers

We will need the following notions of entropy.

Definition 1. *A random variable X has min-entropy k, denoted $\mathsf{H}_\infty(X) \geq k$, if* $\max_x \Pr[X = x] \leq 2^{-k}$.

Next, we define HILL-pseudoentropy, which can be viewed as a "computational version" of min-entropy.

Definition 2 (HILL-pseudoentropy[20, 4]). *A random variable X has* HILL *pseudoentropy k, denoted by $\mathsf{H}^{\mathsf{HILL}}_{\epsilon,s}(X) \geq k$, if there exists a random variable Y with min-entropy $\mathsf{H}_\infty(Y) = k$ where for any distinguisher D of size s*

$$|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| \leq \epsilon$$

In [15], a stream-cipher SC was constructed, which on input a secret initial key $K_0 \in \{0,1\}^\kappa$, generates a stream $X_1, X_2, \ldots$ of pseudorandom blocks $X_i \in \{0,1\}^r$. We write $(K_{i+1}, X_i) \leftarrow \mathsf{SC}(K_i)$. (Here r is a parameter; in our case we need the output length r to be large enough for the key generation algorithm Kg.) The standard property of a stream-cipher is that for any i, the output X_i is pseudorandom given $X_1, \ldots, X_{i-1}$. A leakage-resilient stream-cipher satisfies a much stronger notion as we will explain now.

Let K_{i-1} denote the internal state of SC in round i (right before X_i is computed). Let K^+_{i-1} denote the active state that is accessed by SC in round i to compute X_i. In [15, 37] adversaries are considered which can adaptively sample a leakage function f_i with range $\{0,1\}^\lambda$ before round i, and then obtain $\Lambda_i = f_i(K^+_{i-1})$, the output of the leakage function f_i applied to the entire state that is touched by SC in this round.

It is proven that as long as λ is not too big[14] the output X_i is pseudorandom given the normal output $X_1, \ldots, X_{i-1}$ and the leakage $\Lambda_1, \ldots, \Lambda_{i-1}$. (One can even give the entire state K_i of SC

[14] How large λ can be depends on the security of the underlying pseudorandom generator [15] or function [37].

after X_i was computed, i.e. the cipher is forward secure, but we will not need this.) We will say SC is (s', ϵ', i) secure, if every adversary of size s' has advantage at most ϵ' in distinguishing X_i from random, given $X_1, \ldots, X_{i-1}$ and the leakage $\Lambda_1, \ldots, \Lambda_{i-1}$.

If one is also given the leakage Λ_i, then X_i cannot be pseudorandom any more, as Λ_i could, e.g., be the λ first bits of X_i. But even in this case one can prove [15, Lemma 3] that X_i has high HILL pseudoentropy. In particular, let

$$\mathsf{view}_{q,i} = \{X_1, \ldots, X_{i-1}, \Lambda_1, \ldots, \Lambda_i\}$$

Then if SC is (s', ϵ', i) secure, we get for any $\epsilon_H \geq \sqrt{2^\lambda \epsilon'}$,

$$\Pr_{v:=\mathsf{view}_{q,i}}[\mathsf{H}^{\mathsf{HILL}}_{\epsilon_H, s_H}(X_i|v) = |X_i| - \lambda - 2\log(\epsilon_H^{-1}) - 1] \geq 1 - \epsilon_H \tag{2}$$

where $s_H \approx s' \cdot \epsilon_H^2/8 \cdot |X_i|$.

A.2 SIG^{**}: a Deterministic, Leakage-Resilient Signature-Scheme

Consider the signature scheme SIG^*, but where the randomness for the underlying Kg and Sign queries is generated by a leakage resilient stream cipher SC. We define the signature scheme SIG^{**} almost exactly like SIG^*, except that the randomness for the calls to Kg comes from SC, and thus the entire signing process is deterministic. The scheme is defined in Figure 4. To simplify exposition, we assume that the underlying signature scheme SIG is sateless, and its signing algorithm Sign is deterministic (to achieve this, one can just use any signature scheme, and put the randomness needed to sign three messages into the secret key). In the proof we will also assume, that the entire randomness used to sample $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$ is part of sk. Let us stress that all this is just for exposition, the proof goes through if none of the above holds.

Theorem 2. *Let k be a security parameter, $t = t(k)$,$\epsilon = \epsilon(k) > 0$, and $d, q, \lambda = \lambda(k) \in \mathbb{N}$ (where $q \leq 2^{d+1} - 2$) and $\delta := q \cdot 2^{\lambda+2} \sqrt[3]{\epsilon}$. Further assume the stream cipher SC used in the construction of SIG^{**} is used with a security parameter such that (2) holds with $s_H := t$ and $\epsilon_H := \delta/2q$ for any $i \leq q$. Then if SIG is $(t, 3, \epsilon)$ UF-CMA secure, then SIG^{**} is $(t', q-1, \delta, \lambda)$ UF-CMLA secure where $t' \approx t$.*

Proof. Similar to the proof of Theorem 1, we construct an adversary $\mathcal{F}$ which successfully attacks a $(t, 3, \epsilon)$ UF-CMA secure signature scheme SIG using as a subroutine the adversary $\mathcal{F}_\lambda$ who breaks the UF-CMLA security with leakage λ of SIG^{**}. In the reduction we will loose a factor of at most $q^3 \cdot 2^{3\lambda+4}/\delta^2$, i.e.,

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{SIG}}(\mathcal{F}, k) \geq \frac{\delta^3}{q^3 \cdot 2^{3\lambda+4}}, \qquad \text{where} \qquad \delta = \mathbf{Adv}^{\text{uf-cmla}}_{\mathsf{SIG}^{**}}(\mathcal{F}_\lambda, k). \tag{3}$$

This loss has three reasons: fist, as in Theorem 1, we need to guess the node for which $\mathcal{F}_\lambda$ is going to do a forgery. Second, since we prove the security generically from *any* standard signature scheme we need to guess the leakage (as in Lemma 1). And, finally (and that will be the crucial part) in the real experiment the randomness to sample the keys is output by a stream cipher SC, and thus will only have high HILL-pseudoentropy, whereas in the simulation the target key is sampled uniformly-and-independently. Let us now analyze this more precisely.

Algorithm $\mathsf{Kg}^{**}(1^k)$
$K_{-1} \xleftarrow{\$} \{0,1\}^\kappa$; $(X_0, K_0) \leftarrow \mathsf{SC}(K_{-1})$
$(pk, sk) \leftarrow \mathsf{Kg}(1^k, X_0)$
$S \leftarrow \emptyset$; $\mathsf{push}(S, sk)$; $\Gamma \leftarrow \emptyset$
$SK_0 \leftarrow (w_\varepsilon, S, \Gamma, K_0)$; $PK \leftarrow pk$
return (PK, SK_0)

Algorithm $\mathsf{Vfy}^{**}(PK, m, \Sigma)$
parse Σ 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, Γ, K_i) % 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
$(X_{i+1}, K_{i+1}) \leftarrow \mathsf{SC}(K_i)$ % compute pseudorandom X_i
$(sk_{\hat{w}}, pk_{\hat{w}}) \leftarrow \mathsf{Kg}(1^n, X_{i+1})$ % generate secret key for the current node using X_i
$\sigma \leftarrow \mathsf{Sign}(sk_{\hat{w}}, 1m)$ % 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)
$\phi_{\hat{w}} \leftarrow \mathsf{Sign}(sk_{\mathsf{par}(\hat{w})}, 0pk_{\hat{w}})$ % 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 $\Gamma = \Gamma_{\hat{w}}$
$\Sigma \leftarrow (\sigma, \Gamma)$
$SK_{i+1} \leftarrow (\hat{w}, S, \Gamma, K_{i+1})$ % store key for next signature
return (Σ, SK_{i+1})

Fig. 4. The *deterministic* leakage resilient signature scheme SIG^{**}.

In the attack of SIG, adversary $\mathcal{F}(pk)$ samples an initial key K_{-1}, and then simulates the UF-CMA experiment, where $\mathcal{F}_\lambda$ attacks SIG^{**} instantiated with K_{-1} and is allowed to query for q signatures (and observes their generation). This basically means attacking SIG^*, but where the public/secret key pairs for each node w are sampled as

$$(pk_w, sk_w) \leftarrow \mathsf{Kg}(1^k, X_{\varphi(w)-1}) ,$$

$X_0, X_1, \ldots$ are the output of $\mathsf{SC}(K_0)$.

As in the proof of Theorem 1, $\mathcal{F}(pk)$ makes one exception: for a $\tilde{w}$ sampled at random from the q first nodes (i.e., $\hat{i} \xleftarrow{\$} \{0, \ldots, q\}$ and $\tilde{w} \leftarrow \varphi^{-1}(\hat{i})$), $\mathcal{F}(pk)$ uses its challenge key (pk, sk) as $(pk_{\tilde{w}}, sk_{\tilde{w}})$ instead of sampling it as $(pk_{\tilde{w}}, sk_{\tilde{w}}) \leftarrow \mathsf{Kg}(1^k, X_{\varphi(\tilde{w})-1})$. As in Theorem 1 $\mathcal{F}$ can use its oracle $\mathsf{Sign}(sk, \cdot)$ to compute the signatures, and will guess the leakage $\Lambda_i = f_i(SK_i^+)$ for queries where $sk_{\tilde{w}} \in SK_i^+$ (as in Lemma 1).

We have to lower bound the probability that $\mathcal{F}_\lambda$ will output a forgery for $sk_{\tilde{w}} = sk$. First, as in the proof of Theorem 1, the probability that $\mathcal{F}(pk)$ will correctly guess $\tilde{w}$ (i.e., which of the q keys of SIG adversary $\mathcal{F}_\lambda$ will attack) results in a loss of a factor q. Additionally, when $\mathcal{F}_\lambda$ asks for signatures at nodes $\tilde{w}0$ and $\tilde{w}1$, we loose a factor of $2^{2\lambda}$ for guessing $\Lambda_{\phi(\tilde{w}0)}$ and $\Lambda_{\phi(\tilde{w}1)}$. The loss for visiting $\tilde{w}$ is more subtle. This is because unlike in Theorem 1, in the simulation the randomness X used to sample the challenge key $(pk, sk) \leftarrow \mathsf{Kg}(1^k, X)$ is uniformly random, whereas the replaced key $(pk_{\tilde{w}}, sk_{\tilde{w}}) \leftarrow \mathsf{Kg}(1^k, X_{i+1})$ in the real attack is sampled using a pseudorandom $(X_{i+1}, K_{i+1}) \leftarrow \mathsf{SC}(K_i)$. As a pseudorandom X_{i+1} is indistinguishable from a uniformly random X, this seems not to make much of a difference, but the difficulty is, that the leakage $\Lambda_i = f_i(SK_i^+)$

is not only a function of X_{i+1}, but also of the input $K_i^+ \in SK_i^+$ used by SC to compute the pseudorandom X_{i+1}.

We next prove, that if SC is a sufficiently strong leakage-resilient stream cipher (as required in the statement of the theorem), then we will only loose a factor of $2^{\lambda+2}/\epsilon_H^2$ (where $\epsilon_H := \delta/2q$). This, and the loss of $q \cdot 2^{2\lambda}$ for guessing $\tilde{w}, \Lambda_{\phi(\tilde{w}0)}, \Lambda_{\phi(\tilde{w}1)}$, will give a total loss of $q^3 \cdot 2^{3\lambda+4}/\delta^2$ as claimed in (3).

By assumption (2) is satisfied, thus each X_i has HILL-pseudoentropy $|X_i| - \lambda - 2\log(\epsilon_H^{-1})$ with probability at least $1 - \epsilon_H$ even given $X_1, \ldots, X_{i-1}$ and all leakages $\Lambda_1, \ldots, \Lambda_i$. Taking the union bound over all $i \in [q]$, this will be the case for every $X_i, i \in [q]$ with probability at least

$$\Pr\left[\forall i \in [q] : \mathsf{H}^{\mathsf{HILL}}_{s_H,\epsilon_H}(X_i|X_1, \ldots, X_{i-1}, \Lambda_1, \ldots, \Lambda_i] \geq |X_i| - \lambda - 2\log(\epsilon_H^{-1})\right] \geq 1 - \delta/2 \ . \qquad (4)$$

We say that $\mathcal{F}_\lambda$ in an UF-CMLA attack against SIG^{**} outputs a "good" forgery, if it outputs a forgery, and moreover all X_i have high pseudoentropy as required by (4). As $\mathcal{F}_\lambda$ outputs a (normal) forgery with probability δ, and (4) holds with probability $1 - \delta/2$, $\mathcal{F}_\lambda$ finds a good forgery with probability at least $\delta/2$. By only considering good forgeries, we loose a factor of 2 (notice, that if would not only consider good forgeries, then the adversary could also always attack keys, where the HILL-pseudoentropy of the used randomness was not high).

Now assume that we change the UF-CMLA attack slightly, where we replace the real X_i with an $\tilde{X}_i$ which has min-entropy

$$\mathsf{H}_\infty(\tilde{X}_i|X_1, \ldots, X_{i-1}, \Lambda_1, \ldots, \Lambda_i) \geq |X_i| - \lambda - 2\log(\epsilon_H^{-1}) \ . \qquad (5)$$

By definition of HILL entropy, if (4) holds (and we do not care what happens if not), then there exists such an $\tilde{X}_i$ which cannot be distinguished from X_i with advantage more than ϵ_H. By doing this transition we loose a factor of $1/(1 - \epsilon_H) < 2$.

Consider any random variable $\tilde{X}$ where $\mathsf{H}_\infty(\tilde{X}) = |\tilde{X}| - \alpha$. By Lemma 4 in Appendix A.3, there exists an event $\mathcal{E}$, having probability at least $2^{-\alpha}$, such that a uniformly random X conditioned on $\mathcal{E}$ has the same distribution as $\tilde{X}$. In particular, with (5) this implies that we can define an event $\mathcal{E}$ where $\Pr[\mathcal{E}] \geq 2^{-\lambda}\epsilon_H^2$, such that conditioned on that event, the uniformly random X used to sample the challenge key (pk, sk) has the same distribution as $\tilde{X}_i$, and thus conditioned on this event, we have perfectly simulated the i-th query in the UF-CMLA attack (with $\tilde{X}_i$ instead of X_i as considered above). Here we loose a factor of $2^\lambda/\epsilon_H^2$. Putting the factors together we get the claimed bound. □

A.3 A technical lemma

Lemma 4. *Consider any random variable $\tilde{X} \in \{0,1\}^r$ where $\mathsf{H}_\infty(\tilde{X}) = r - \alpha$, and let X be uniform over $\{0,1\}^r$. Then there exists an event $\mathcal{E}$ such that*

$$\Pr[\mathcal{E}] = 2^{-\alpha} \qquad \textit{and} \qquad \forall x : \Pr[X = x|\mathcal{E}] = \Pr[\tilde{X} = x] \ .$$

Proof. We define the event $\mathcal{E}$ (depending on X) as

$$\Pr[\mathcal{E}|X = x] = \Pr[\tilde{X} = x]/2^{\alpha - r} \ .$$

The probability of $\mathcal{E}$ is

$$\Pr[\mathcal{E}] = \sum_{x \in \{0,1\}^r} \Pr[X = x] \cdot \Pr[\mathcal{E}|X = x] \tag{6}$$

$$= \sum_{x \in \{0,1\}^r} 2^{-r} \cdot \Pr[\tilde{X} = x]/2^{\alpha - r} = 2^{-\alpha} \, . \tag{7}$$

Further, conditioned on $\mathcal{E}$, X has the same distribution as $\tilde{X}$:

$$\Pr[X = x|\mathcal{E}] = \frac{\Pr[X = x \wedge \mathcal{E}]}{\Pr[\mathcal{E}]} = \frac{\Pr[X = x] \cdot \Pr[\mathcal{E}|X = x]}{\Pr[\mathcal{E}]} = \Pr[\tilde{X} = x] \, .$$

B Illustration of the leakage resilient signature scheme SIG*

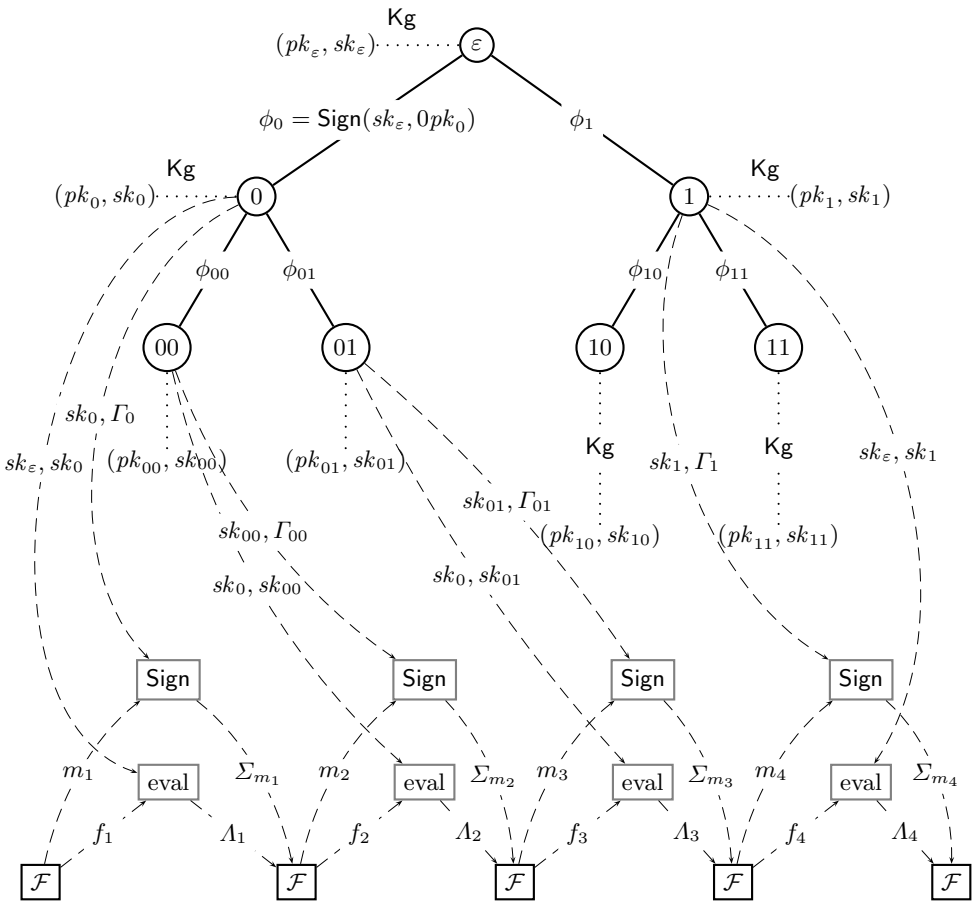

Fig. 5. Illustration of the execution of SIG^* in the UF-CMLA experiment. This figure shows the first 4 rounds of interaction between the adversary $\mathcal{F}$ and Sign. The dotted edges associate a public/secret key to each node. The dashed arrows represent $\mathcal{F}$'s oracle queries. We omit some technical details. $\mathcal{F}$ 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.